Politechnika Warszawska

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

Praca dyplomowa magisterska

na kierunku Inżynieria i analiza danych

Zastosowanie metod analityki danych do analizy zachowań zespołu serwerów

Damian Rakus

Numer albumu 286395

promotor

dr hab. prof. IBS PAN Marcin Paprzycki

WARSZAWA 2021

....

podpis promotora podpis autora

Streszczenie

Zastosowanie metod analityki danych do analizy zachowań zespołu serwerów

W dzisiejszych czasach drastycznie rośnie liczba serwerów używanych do hostowania przeróżnych aplikacji. W celu nadzorowania stanu tych serwerów używane są logi, które zawierają metryki serwerów w danym momencie w czasie. Jednakże wraz z rozwojem branży IT, infrastruktura rozwiązań staje się coraz bardziej skomplikowana, co prowadzi do zwiększenia liczby parametrów raportowanych w logach do tego stopnia, że monitorowanie stanu systemu przez człowieka w czasie rzeczywistym staje się niemożliwe. Jedynym rozwiązaniem problemu jest zlecenie zadania nadzorowania infrastruktury komputerom.

W niniejszej pracy podjęto się stworzenia modeli, które wykrywałyby anomalie na podstawie logów zawierających dane wielowymiarowe. Ich cechami wyróżniającymi są: użycie warstw self attention oraz zastosowanie autorskiego podejścia zwanego statystyką zmian w celu redukcji liczby fałszywych pozytywów. Dodatkowo opracowany został dedykowany sposób oceny modeli w zadaniu wykrywania anomalii, który uwzględnia fakt, że logi mogą zawierać niestandardowe wartości kilka chwil przed lub po awarii systemu.

W pierwszym rozdziale została opisana tematyka problemu, którą obejmuje praca magisterska. W kolejnej sekcji pracy przeanalizowane zostały podejścia, znalezione w literaturze, które były używane przez badaczy w dziedzinach zbliżonych do tematyki problemu oraz bliżej opisane zostały wybrane metody zarówno pojawiające się w pracach innych naukowców jak i te, które użyte zostały w ramach pracy magisterskiej. W dalszym rozdziale opisane zostały dane wykorzystane w pracy magisterskiej oraz wykorzystane podejścia, a także zdefiniowano miary oceny modeli. Następnie przedstawione zostały wyniki badań prowadzonych w ramach pracy magisterskiej. Ostatni rozdział zawiera ocenę końcowych rezultatów oraz wymienia dalsze możliwości rozwoju badań nad tematem pracy. W dodatku natomiast zawarte zostały rysunki architektur sieci neuronowych użytych w pracy.

Słowa kluczowe: Wykrywanie anomalii, sieci neuronowe, metody uczenia głębokiego, serie czasowe, AutoEncodery, LSTM, CNN, modele głosujące

Abstract

Applying data analytics to data originating from a farm of servers

Nowadays, a number of servers that are used to host various applications drastically increases. In order to oversee the state of those servers, we store their metrics at a given timestamp in log files. However, along with development of IT sector, the infrastructure of software solutions is becoming more complex causing the number of parameters stored in log files to rise to the point that it is impossible for humnan to monitor system's state in real time. The only solution to this problem is to delegate the task of infrastructure monitoring to the machines.

The goal of master thesis is to create machine learning models that would detect anomalies based on log file containing multiple parameters. Their distinct features comprise of usage of self attention layers and application of custom approach called "Variation statistic" that lowers the count of false positives reported by models. Moreover, thesis includes novel benchmark method dedicated specifically for evaluation of anomaly detection models which takes into account the fact that log files can contain outliers a few moments before or after critical system malfunction.

The first chapter contains description of the problem, which thesis aims to solve. In next section, approaches applied by researchers in similiar problems, that are described in publicly available literature, are analysed. In addition methods used both by researchers and in master thesis were described in detail. Next chapter contains description of datasets used in thesis along with approaches applied. In addition, metrics for model evaluation were defined. Next, results of tests conducted were shown and explained. Last chapter contains evaluation of final results and possibilites of further research development of thesis subject. In appendices, figures presenting architectures of neural networks used in thesis are attached.

Keywords: Anomaly detection, neural networks, deep learning, time series, AutoEncoders, LSTM, CNN, ensemble models

Warszawa, dnia .....

Oświadczenie

Oświadczam, że pracę magisterską pod tytułem "Zastosowanie metod analityki danych do analizy zachowań zespołu serwerów", której promotorem jest dr hab. prof. IBS PAN Marcin Paprzycki, wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.

.....

author's signature

Spis treści

1. Wstęp

Żyjemy obecnie w latach transformacji cyfrowej, coraz więcej branży zaczyna wdrażać rozwiązania IT w swoich przedsiębiorstwach. Coraz więcej ludzi korzysta z usług oferowanych przez firmy w internecie np. poprzez robienie zakupów online, zlecanie przelewów internetowych, czy oglądając filmy na platformach takich jak HBO GO czy Netflix. Wszystkie te usługi są udostępniane za pośrednictwem przeróżnych aplikacji hostowanych na serwerach. Firmy inwestują duże pieniądze, by zapewnić dostępność ich usług 24 godziny na dobę, a wyspecjalizowane zespoły są odpowiedzialne za nadzorowanie stanu serwerów, które udostępniają oferowane serwisy. Jakakolwiek awaria musi zostać jak najszybciej wykryta oraz naprawiona, gdyż długa przerwa w działaniu aplikacji niesie za sobą utratę przychodów oraz stratę zaufania klientów, którzy następnie zaczną rozważać korzystanie z usług gwarantowanych przez konkurenta.

Jednakże monitorowanie tych systemów staje się coraz trudniejsze. Wraz z rozwojem technologii oraz zwiększaniem wachlarza usług przez firmy rośnie złożoność infrastruktury IT. W celu ujednolicenia sposobu oceny stanu serwera wykorzystywany jest mechanizm logowania do pliku, w którym zapisywane są parametry serwera w danej chwili. Jednakże liczba parametrów jest tak duża, że człowiek nie potrafiłby wynieść ważnych informacji poprzez czytanie pliku tekstowego z logami. Na pomoc przychodzą wtedy dedykowane rozwiązania, których zadaniem jest agregacja logów, przedstawianie kluczowych statystyk w formie wykresów oraz częściowe ostrzeganie o potencjalnych problemach. Niestety, z czasem ogrom przychodzących danych stał się tak duży, że nawet oprogramowanie zapewniające formę wizualizacji parametrów występujących w logach, nie zapewnia wystarczającej przejrzystości, żeby człowiek mógł w czasie rzeczywistym nadzorować stan całej infrastruktury serwerowej. Ostatnią deską ratunku zostało oddelegowanie zadania nadzoru nad infrastrukturą IT dedykowanemu oprogramowaniu.

W tym celu zgłosiła się do nas firma EMCA. Oferuje ona oprogramowanie zwane Energy Logserver, które zajmuje się agregacją logów pochodzących z wielu urządzeń. Zaimplementowane przez nich rozwiązanie informuje o awarii, jeżeli co najmniej jeden z określonych parametrów przekroczy ustaloną wartość krytyczną. Jest to jednak duże uproszczenie, gdyż nie każde przekroczenie tej wartości jest anomalią oraz nie każda anomalia wyróżnia się przekroczeniem wartości krytycznej. Niektóre anomalie mogą wynikać z tego, że zależność pomiędzy pewnymi

parametrami została zaburzona. Dodatkowo podczas działania systemu można wyróżnić pewne okresy, w których przekroczenie wartości krytycznych nie jest czymś niespotykanym. Właśnie dlatego podejście naiwne oparte na wartościach krytycznych nie jest idealne, co niesie za sobą potrzebę powstania metody, która będzie w lepszy sposób potrafiła zidentyfikować anomalie.

Właśnie z tym problemem, zmierzono się w pracy magisterskiej. Celem pracy magisterskiej jest stworzenie modelu do wykrywania anomalii, który będzie wykorzystywał metody uczenia maszynowego. Do zadań rozwiązanie należą:

- wykrywanie anomalii występujących w systemie na podstawie logów,
- przetwarzanie serii czasowych zawierających wiele zmiennych,
- uczenie się w trybie nienadzorowanym, gdyż dane pochodzące z logów nie są etykietowane.

Zadaniem detekcji anomalii na podstawie danych firmy EMCA zajmowano się również w pracy [9]. Jednakże w tamtym przypadku dane pochodziły z innego programu, a także użyto metod uczenia maszynowego, które nie radzą sobie z danymi wielowymiarowymi. Zatem w niniejszej pracy zastosowano całkowicie inne podejście, które pozwala między innymi na przetwarzanie danych wielowymiarowych.

2. Przegląd metod oraz literatury

W przeciągu ostatnich kilkunastu lat ukazało się wiele publikacji naukowych dotyczących wykrywania anomalii. Niektóre z nich opierały się na danych pochodzących z serwerów, co jest bliskie problematyce niniejszej pracy magisterskiej. Autorzy badań proponują wiele modeli, które mogą okazać się skuteczne w zadaniu oraz porównują ich skuteczność z innymi metodami predykcji serii czasowych. Poniżej omówione zostały metody wykorzystywane przez badaczy w zadaniu wykrywania anomalii. Są to między innymi podejścia oparte na metodzie PCA, metodzie Holta-Wintersa, modelu ARIMA oraz jego modyfikacjach, a także sieciach neuronowych. Ich wykorzystanie pozwoliło badaczom na skuteczne wykrycie anomalii w użytych zbiorach danych.

2.1. Analiza składowych głównych (ang. PCA – Principal Component Analysis)

2.1.1. Opis metody

Metoda PCA – jest to jedna z metod analizy czynnikowej. Polega ona na redukcji wymiaru danych poprzez zastąpienie oryginalnych cech nowymi, które tłumaczą jak najwięcej zmienności w danych. Nowe cechy nazywane są składowymi głównymi.

Oznaczając $x = (x_1, x_2, ..., x_k)'$ – wektor oryginalnych cech.

Celem jest wyznaczyć takie cechy $y_1, ..., y_k$, które tłumaczą jak najwięcej zmienności w danych oraz są kombinacją liniową oryginalnych cech tj. $y_i = a_i x$, gdzie $a_i \in R^p$.

Chcąc tłumaczyć jak najwięcej zmienności danych trzeba znaleźć wektory $a_1, ..., a_k$ tak, żeby maksymalizowały one wariancję danych tzn.

$$\begin{aligned}
a_1 &= arg\max_{a \in R^p} Var(a'x), ||a|| = 1 \\
a_2 &= arg\max_{a \in R^p} Var(a'x), ||a|| = 1, a'a_1 = 0 \\
&... \\
a_i &= arg\max_{a \in R^p} Var(a'x), ||a|| = 1, \forall_{j<i}\, a'a_j = 0 \\
&... \\
a_k &= arg\max_{a \in R^p} Var(a'x), ||a|| = 1, \forall_{j<k}\, a'a_j = 0
\end{aligned}$$

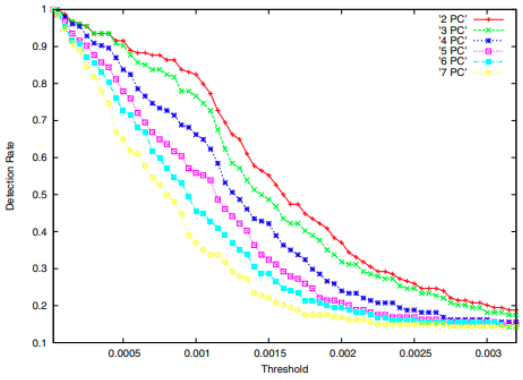
Rozwiązaniem powyższego układu równań są wektory własne $w_1, ..., w_k$ odpowiadające wartościom własnym $\lambda_1 \geqslant \lambda_2 \geqslant .. \geqslant \lambda_k$ macierzy kowariancji, w której kolumnom odpowiadają wektory cech.

Zatem metodę PCA można opisać w następujących krokach:

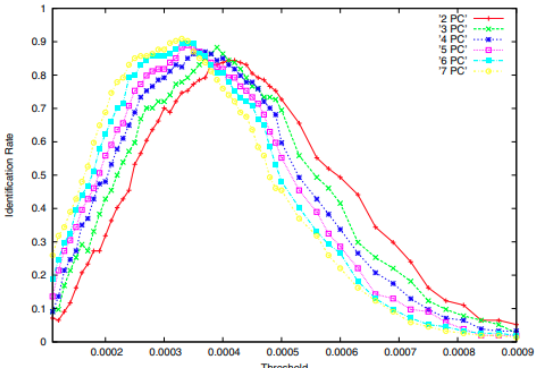
1. Skonstruuj macierz X o wymiarach $n \times k$ tak, że kolumnom odpowiadają wektory cech $x_1, x_2, .., x_k$
2. Znormalizuj macierz X tak, by wartości w kolumnach miały średnią równą 0 oraz wariancję równą 1. Ten krok jest wykonywany w celu uzyskania tej samej skali dla wszystkich cech.
3. Oblicz macierz kowariancji $cov(X)$
4. Wyznacz wektory własne $w_1, w_2, .., w_p$ oraz odpowiadające im wartości własne $\lambda_1, \lambda_2, ..., \lambda_p$
5. Na podstawie wartości własnych wybierz $l(l < k)$ wektorów, które tłumaczą jak najwięcej zmienności w danych, według ustalonego progu θ (np. $\theta = 0.9$)
6. Oblicz składowe główne $y_1 = Xw_1, y_2 = Xw_2, ..., y_l = Xw_l$

2.1.2. Zastosowanie metody

Metodę PCA wykorzystano między innymi w pracy [2]. Technika ta została użyta w celu znajdowania anomalii w sieci internetowej. Dane, na których przeprowadzone były badania to ruch sieciowy w architekturze Abilene/Internet2, która jest siecią używaną przez społeczność badaczy na terenie Stanów Zjednoczonych Ameryki. Dane na temat ruchu zawierają informacje na temat liczby pakietów wysyłanych pomiędzy 9 ruterami i były zbierane na przestrzeni jednego tygodnia. Dodatkowo badacze wzbogacili dane o 155 anomalii, które miały symulować atak DoS (Denial of Service). Następnie dane zostały posortowane i podzielone w okna czasowe tak, by utworzyć serię czasową. Potem badacze zastosowali metodę PCA i podjęli następujące założenie: dominujące składowe główne (czyli takie, które tłumaczą określoną przez pewien próg zmienność) uznane zostały za składowe opisujące przestrzeń normalną, natomiast pozostałe składowe reprezentują przestrzeń anomalii. W dalszej kolejności za pomocą niedominujących składowych głównych przedział czasowy był rzutowany jako wektor anomalii. Jeśli druga norma wektorowa nałożona na ten wektor przekraczała określony przez badaczy próg, to przedział czasowy był oznaczany jako przedział, w którym wydarzyła się anomalia. Autorom pracy udało się przy odpowiednim doborze liczby dominujących składowych głównych oraz wartości progu występowania anomalii wykryć około 90% anomalii występujących w systemie. Dodatkowo prawie 90% anomalii zgłaszanych przez system było rzeczywistymi anomaliami, co widać na wykresach 2.1



(a) Wykres informujący jaka część rzeczywistych anomalii została zgłoszona przez system była rzeczywistymi anomaliami w zależności od progu występowania anomalii dla modeli o różnej ilości składowych głównych.



(b) Wykres informujący jaka część anomalii zgłoszonych przez model była rzeczywistymi anomaliami w zależności od progu występowania anomalii dla modeli o różnej ilości składowych głównych.

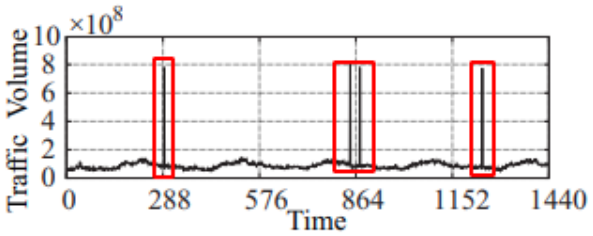
Rysunek 2.1: Wyniki metody PCA. (Źródło: [2])

Metoda PCA nie jest jednak idealna jak pokazują naukowcy w pracy [10]. Jeśli anomalie osiągają wartości znacznie większe od standardowych wartości, to dominujące składowe główne przekształcają anomalie na przestrzeń normalną przez co anomalie nie zostają wykryte. W związku z tym badacze zaproponowali metodę, która we wstępnym przetwarzaniu danych pozbywa się wartości odstających na podstawie okresowych (w tym przypadku tygodniowych) wzorców w ruchu sieciowym. Dzięki temu, anomalie o dużych wartościach wykrywane są zanim zastosowania będzie metoda PCA, co chroni przed problemem zaburzenia przestrzeni normalnej. Na wykresach 2.2 można zobaczyć jak różnią się rzutowania ruchu sieciowego w wariancie klasycznego PCA oraz PCA ze wstępnym odrzuceniem wartości odstających.

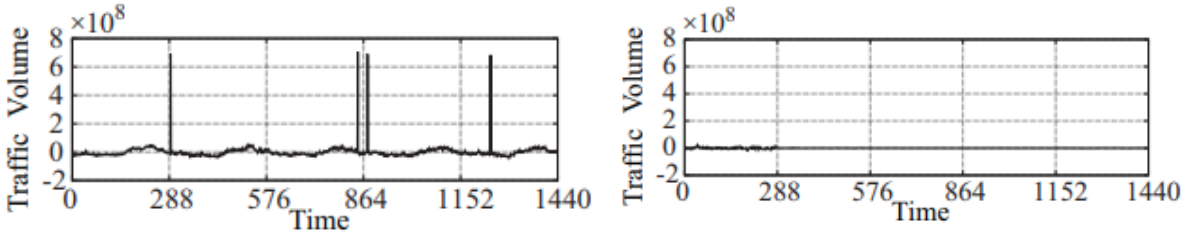
Naukowcy następnie przeprowadzili porównanie klasycznej metody PCA oraz jej modyfikacji. Użyli danych, które również pochodzą z sieci Abilene, do których dodawali anomalie o różnej wartości ruchu sieciowego. Pokazali, że gdy anomalie mają dużą wartość to klasyczna metoda PCA nie radzi sobie z ich wykrywaniem, co prowadzi do dużej liczby fałszywych negatywów, natomiast ich metoda poprawnie identyfikowała te anomalie. Warto również zaznaczyć, że jeśli anomalie nie przyjmują dużych wartości to obie metody działają podobnie. Wyniki badań widoczne są na Rysunku 2.3.

Krytyka metody PCA została również wyrażona w pracy [16]. Jako główne wady tej techniki wymieniono:

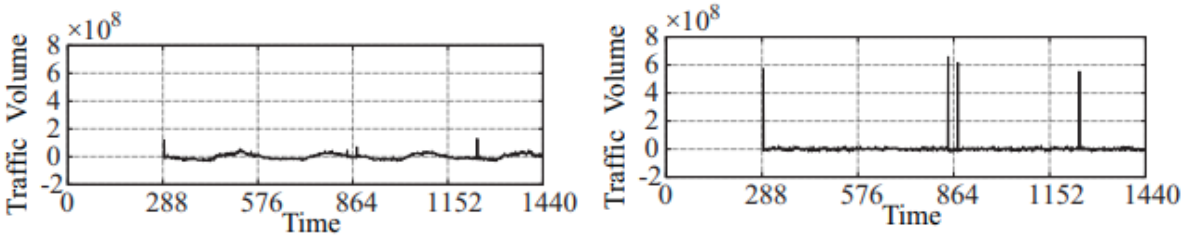
- Wskaźnik fałszywych pozytywów jest wysoce zależny od liczby wybranych dominujących



(a) Wykres przedstawiający ruch sieciowy z zaznaczonymi na czerwono anomaliami



(b) Klasyczna metoda PCA - po lewej przestrzeń normalna, na której widać anomalie, natomiast po prawej przestrzeń anomalii, która nie zawiera anomalii



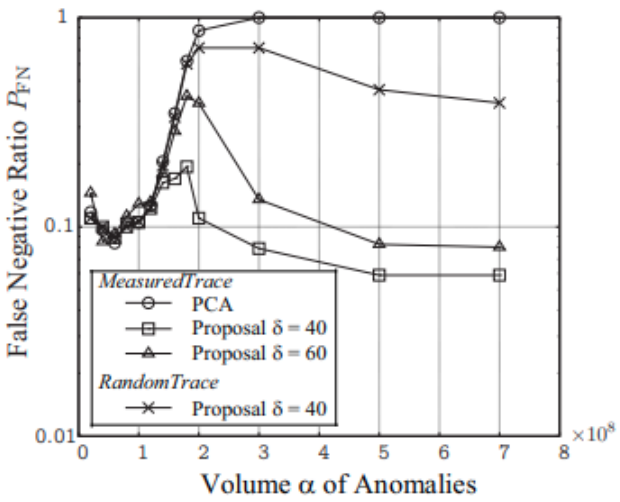
(c) Zmodyfikowana metoda PCA – po lewej przestrzeń normalna już bez anomalii, po prawej anomalie zostały poprawnie zmapowane na przestrzeń anomalii

Rysunek 2.2: Wpływ dużych anomalii na klasyczną oraz zmodyfikowaną metodę PCA. (Źródło: [10])

składowych głównych w metodzie PCA. Według badaczy zależnie od ich liczby wskaźnik fałszywych pozytywów może się różnić trzykrotnie. W związku z tym dominujące składowe główne powinny być dobierane na podstawie testów skuteczności wykrywania anomalii dla każdego zagadnienia oddzielnie.

- Skuteczność metody PCA jest zależna od stopnia agregacji danych. Jeśli dane są zbyt mocno zagregowane, to metoda nie będzie w stanie wykrywać anomalii oprócz znacznie różniących się od wartości normalnych, jeśli jednak stopień agregacji będzie za mały, to wtedy przedziały czasowe będą się bardzo różnić, co spowoduje, że metoda będzie zgłaszać wiele fałszywych pozytywów.

- Anomalie o dużych wartościach zanieczyszczają przestrzeń normalną, przez co anomalie nie są wykrywane. Z tym problemem można jednak sobie poradzić, przetwarzając wstępnie



Rysunek 2.3: Wykres prezentujący wskaźnik fałszywych pozytywów w zależności od wielkości anomalii dodanych do zbioru danych. Linia z kółkami reprezentuje metodę PCA, natomiast pozostałe linie reprezentują zmodyfikowaną metodę PCA z różnymi progami wykrywania wartości odstających. (Źródło: [10])

dane i odrzucając wartości odstające, co uczyniono w pracy [10].

- O ile wykrycie występowania anomalii nie jest problemem, to ciężko jest zinterpretować co ją spowodowało. Anomalie są wykrywane na podstawie danych, które są rzutem danych rzeczywistych na przestrzeń anomalii. Nie ma jednak bezpośredniego mapowania z przestrzeni anomalii do danych rzeczywistych.

2.2. Suma kumulatywna po wielu zmiennych – MCUSUM

2.2.1. Opis metody

MCUSUM jest to sposób liczenia sumy kumulatywnej dla wektora zmiennych. Kumulatywna suma błędu predykcji po wielu zmiennych wartości jest używana w metodach wykrywania anomalii. Zakłada się w niej, że wystąpiła anomalia, jeśli wartość sumy przekroczyła pewną graniczną wartość.

Wartość sumy kumulatywnej wielu zmiennych w chwili t oznacza się jako C_t i wylicza się następująco przy założeniu, że wartości X_i, dla których obliczana jest suma kumulatywna mają rozkład normalny o średniej μ_0 oraz wariancji σ^2:

$$C_t = \sqrt{(L_{t-1} + X_t + \mu_0)^T \Sigma^{-1} (L_{t-1} + X_t + \mu_0)},$$

gdzie

$$L_t = \begin{cases} 0 & \text{jeśli} \quad C_t \leqslant k \\ (L_{t-1} + X_t - \mu_0)(1 - \frac{k}{C_t}) & \text{w pp.} \end{cases}$$

$$L_0 = 0$$

$$k = \frac{\sqrt{(\mu_1 - -\mu_0)^T \Sigma^{-1} (\mu_1 - -\mu_0)}}{2}$$

Σ – macierz kowariancji dla macierzy X, w której X_i jest i-tą kolumną

$\mu_1 = \mu_0 + \delta\sigma, \delta$ – pewien wybrany wektor

2.2.2. Zastosowanie metody

Ze względu na wymienione w sekcji 2.1.2 wady metody PCA, istnieje potrzeba jej udoskonalenia. Jednym z przykładów modyfikacji tej metody jest technika opisana w pracy [16], która opiera się na wykrywaniu anomalii za pomocą progu dla sumy kumulowanej wielu zmiennych (ang. Multivariate Cumulative Sum). Metoda ta polega na tym, że na danych wejściowych stosuje się metodę PCA, a następnie na podstawie wyznaczonego przez nią rzutowania danych na przestrzeń anomalii został wyznaczony próg dla sumy kumulowanej wielu zmiennych po przekroczeniu którego system zgłaszał anomalię.

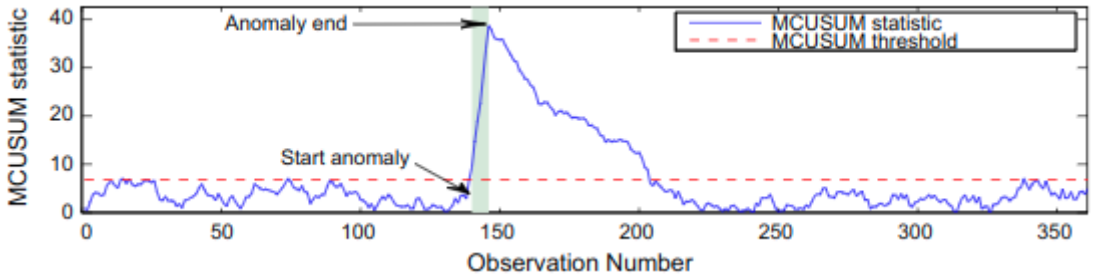
Badacze porównali ich metodę z innymi wariantami PCA takimi jak metoda oparta na Statystyce Q oraz Statystyce T^2 Hotellinga. Testy zostały przeprowadzone na danych pochodzących z oddziału ratunkowego szpitala w francuskiej miejscowości. Dane zbierane były na przestrzeni jednego roku i zawierają takie cechy jak:

- dzienna liczba pacjentów przywiezionych na oddział ratunkowy przez karetkę,
- dzienna liczba pacjentów, którzy sami przyjechali na oddział ratunkowy,
- dzienna liczba pacjentów w stanie ciężkim,
- dzienna liczba pacjentów w stanie stabilnym,
- dzienna liczba niespodziewanych przybyć pacjentów,
- dzienna liczba pacjentów na oddziale radiologii,
- dzienna liczba pacjentów zbadanych przez skaner,
- dzienna liczba pacjentów, którym wykonano echografię,
- dzienna liczba pacjentów na badania biomedyczne,

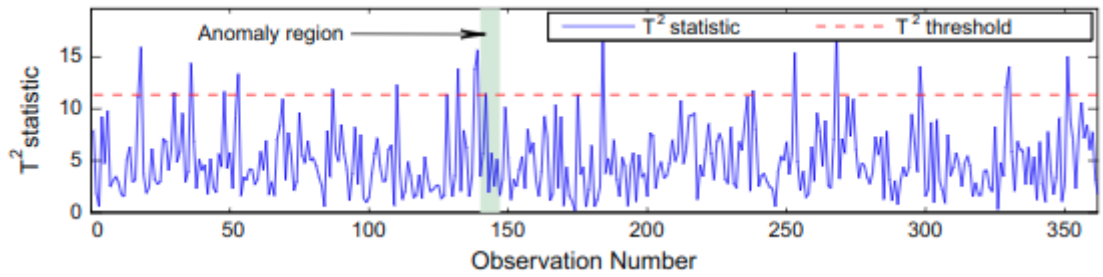
- dzienna liczba wypisanych pacjentów.

Dane jednak nie zawierały anomalii, więc naukowcy dodali anomalię poprzez dodanie zmiany wynoszącej 25% całkowitej wariancji dla danych pomiędzy 141. a 147. dniem.

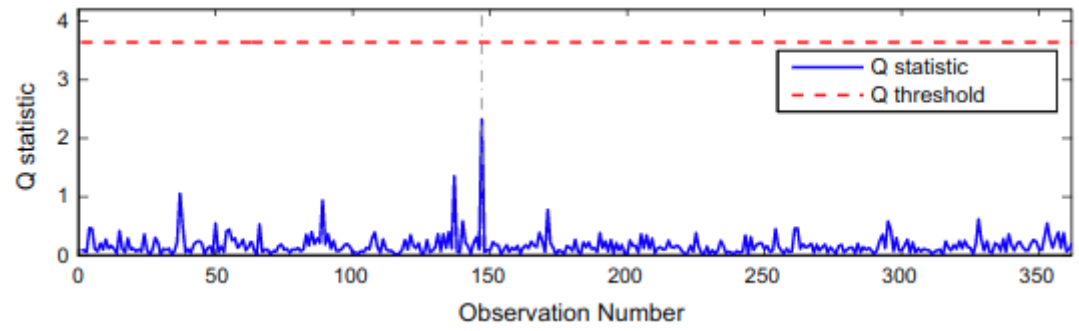
Na wykresach 2.4 zaprezentowanych przez badaczy widać, że metoda oparta na kumulatywnej sumie wielu zmiennych poradziła sobie bardzo dobrze, gdyż wykryła dodaną sztucznie anomalię, chociaż raportowana długość jej trwania była znacząco dłuższa niż rzeczywista. Wykrywanie anomalii zgodnie z Statystyką T^2 Hotellinga nie znalazło anomalii i dodatkowo zgłosiło anomalie w miejscach, w których anomalii nie było (tzw. fałszywe pozytywy). Natomiast wykrywanie anomalii zgodnie z statystyką Q nie zgłosiła żadnych anomalii.



(a) Metoda PCA oparta na sumie kumulatywnej wielu zmiennych



(b) Metoda PCA z Statystyką T^2 Hotellinga



(c) Metoda PCA z Statystyką Q

Rysunek 2.4: Porównanie metody PCA opartej na sumie kumulatywnej wielu zmiennych z standardowymi metodami PCA. (Źródło: [8])

Metoda ta z racji, że wykorzystuje sumę kumulatywną wielu zmiennych ma jednak wady.

Po pierwsze o ile metoda ta z łatwością identyfikuje długotrwałe anomalie, to miewa problemy z wykryciem nagłych, krótkotrwałych anomalii, gdyż w krótkim odstępie czasu, suma kumulatywna może nie przekroczyć wartości granicznej. Dodatkowo z racji, że żeby suma kumulatywna przekroczyła wartość krytyczną wymagane jest kilka pomiarów, to anomalia wykrywana jest z lekkim opóźnieniem.

2.3. RPCA – Robust Principal Component Analysis

2.3.1. Opis metody

Jedną z wady metody PCA jest fakt, że algorytm jest czuły na obserwacje odstające *(ang. outliers)*. W zagadnieniu wyszukiwania anomalii obserwacjami odstającymi są właśnie anomalie. W przypadku, gdy w zbiorze danych występuje dużo anomalii, to metoda PCA wyliczy składowe główne z dużym błędem, gdyż algorytm będzie próbował znaleźć jak najlepsze mapowania zarówno dla obserwacji poprawnych jak i obserwacji błędnych. W tym celu powstała metoda RPCA, będąca modyfikacją metody PCA, która jest odporna na uszkodzone dane *(ang. corrupted data)*.

W metodzie RPCA macierz danych X z dużą ilością anomalii można zapisać jako sumę $X = L_0 + S_0$, gdzie L_0, to macierz o małym rzędzie, która intuicyjnie będzie reprezentować poprawne obserwacje, a S_0, to macierz rzadka, która intuicyjnie będzie zawierać anomalie.

Wydawać się może, że to zadanie jest niemożliwe, gdyż brakuje informacji jak szukać macierzy L_0 oraz S_0. Okazuje się jednak, że można je otrzymać rozwiązując następujący problem optymalizacyjny:

$$\text{zminmalizuj} \quad ||L||_* + \lambda||S||_1$$

$$\text{pod warunkiem, że} \quad L + S = X$$

$$\text{gdzie} \quad ||X||_* := \sum_i \sigma_i(X) - \text{(ang. nuclear norm)}$$

$$||M||_1 := \sum_{ij} |X_{ij}|$$

2.3.2. Zastosowanie metody

Metodę RPCA, będącą udoskonaleniem metody PCA, z powodzeniem użyto w pracy [14]. Zastosowano tam założenie, że macierz danych można rozłożyć na sumę macierzy L – o małym rzędzie oraz macierzy rzadkiej S. Metoda ta była zastosowana na danych sieciowych zawierających takie informacje jak:

- adres IP nadawcy oraz odbiorcy pakietu,

- port nadawcy oraz odbiorcy pakietu,
- protokół użyty do wysłania pakietu,
- wielkość pakietu,
- czas wysyłania pakietu.

Dane zostały podzielone na 3 części z czego każda zawierała inny rodzaj ataku hakerskiego:

1. atak ICMP sweep – polegający na zbadaniu jakie adresy są przypisane do maszyn,
2. próba wyszukania pewnego procesu demona (ang. daemon) – czyli procesu działającego w tle na pewnej maszynie,
3. próba wykorzystania luki bezpieczeństwa w znalezionym w poprzedniej części procesie.

Pierwsze dwa zbiory danych służyły za dane treningowe, natomiast trzeci zbiór danych służył za zbiór testowy w celu sprawdzenia czy metoda RPCA rozpozna typ ataku, który nie występował w zbiorze treningowym. Badacze wykazali, że metoda RPCA poradziła sobie lepiej od tradycyjnej metody PCA w wykrywaniu nowego typu ataku hakerskiego, ponieważ miała wyższy wskaźnik prawdziwych pozytywów oraz rzadziej zgłaszała fałszywy alarm. Dodatkowo badacze pokazali, że dobór parametru λ występującego w metodzie PCA oraz progu zgłaszania anomalii za pomocą metody sprawdzianu krzyżowego przynosi lepsze rezultaty niż wybór wartości tych parametrów na podstawie wiedzy teoretycznej.

Metoda ta jednak nie jest skuteczna, gdy dane na których jest zastosowana wychodzą poza zakres wartości obecnych w zbiorze treningowym. Przykładowo gubi się ona w przypadku, jeżeli wartości danych stale rosną w czasie. Taka sytuacja zdarza się często, gdy dana usługa zyskuje na popularności i z tego powodu serwer staje się bardziej obciążony. Żeby model oparty na RPCA przystosował się do takich zmian wartości w logach, musi on zostać wytrenowany od nowa.

2.4. Metoda sprawdzianu krzyżowego z oknem ruchomym

2.4.1. Opis metody

Metoda ta jest odmianą sprawdzianu krzyżowego dla serii czasowych. Polega ona na zbadaniu jak dana metoda sobie radzi z predykcją serii czasowych w zależności od liczności zbioru treningowego.

Algorytm tej metody można opisać następująco:

1. Ustal liczbę obserwacji p, która będzie wchodzić w skład zbioru testowego jaka część zbioru danych będzie zbiorem testowym, np. $p = 0.2 \times$ liczba wszystkich obserwacji w zbiorze danych.

2. Za zbiór treningowy weź początkowe p obserwacji z zbioru danych, kolejne p obserwacji występujących bezpośrednio po danych treningowych weź jako zbiór testowy.

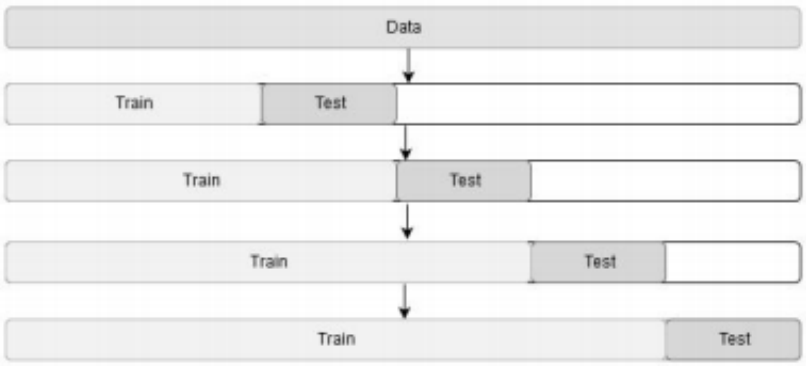
3. Przeprowadź trening, a na zbiorze testowym oblicz skuteczność modelu.

4. Zwiększ zbiór treningowy o zbiór testowy, za zbiór testowy weź kolejne p obserwacji bezpośrednio po zbiorze treningowym. Jeśli nie ma już więcej danych, które można przypisać do zbioru treningowego to przejdź do punktu 6.

5. Wróć do punktu 3.

6. Oblicz skuteczność modelu jako średnią skuteczność modelu w każdej iteracji testowej i zakończ algorytm.

Zwizualizowany przykład podziału zbioru na zbiór treningowy oraz testowy w metodzie sprawdzianu krzyżowego z ruchomym oknem można zobaczyć na Rysunku 2.5



Rysunek 2.5: Podział zbioru danych metodą sprawdzianu krzyżowego z ruchomym oknem Źródło: Artykuł [5]

2.4.2. Zastosowanie metody

Metoda sprawdzianu krzyżowego z oknem ruchomym jest wykorzystywana przy podziale zbioru danych w celu oceny modelu. Została ona przykładowo wykorzystana, żeby ocenić skuteczność metody Holta-Wintersa w pracy [5], która została dokładniej opisana w rozdziale 2.5.2.

2.5. Metoda potrójnego wygładzania wykładniczego (metoda Holta-Wintersa)

2.5.1. Opis metody

Jest to metoda używana w przewidywaniu serii czasowych. Polega ona na przypisywaniu wykładniczo rosnących wag danym z serii oraz braniu pod uwagę sezonowości danych (czyli powtarzalności jakiegoś wzorca w danych w czasie).

Zastosowano następujące oznaczenia:

- x_t – wartość danych w momencie t,
- R_t – wykładniczo wygładzona wartość przewidywanej zmiennej w chwili t,
- T_t – wartość przyrostu trendu w chwili t,
- S_t – wskaźnik sezonowości w chwili t,
- L – długość cyklu sezonowości,
- α, β, γ – parametry modelu o wartościach z przedziału [0,1],
- $m, m < L$ – dla ilu kroków do przodu przewidywana jest wartość danych,
- $\bar{x}_{t+m}$ – przewidywana wartość w chwili $m + t$ na podstawie rzeczywistej wartości danych w chwili t.

W tej metodzie na początku czeka się, aż zostaną zebrane dane z początkowych N cykli, z czego $N >= 2$. Następnie inicjuje się początkowe wartości jako

$$\begin{aligned} R_0 &= x_0 \\ T_0 &= \frac{1}{L}\left(\frac{x_{L+1}-x_1}{L} + ... + \frac{x_{L+L}-x_L}{L}\right) \\ S_i &= \frac{1}{N}\sum_{j=1}^{N} \frac{x_{L(j-1)+i}}{A_j} \text{ dla i=1,2,...,L, gdzie} \\ A_j &= \frac{\sum_{i=1}^{L} x_{L(j-1)+i}}{L} \text{ dla j=1,2,...,N} \end{aligned}$$

Natomiast parametry α, β, γ są inicjalizowane losowymi wartościami z przedziału [0, 1]

Zatem można zaobserwować, że początkowa wygładzona wartość danych, to wartość początkowa. Początkowa wartość trendu, to średnia zmian pomiędzy wartościami w odpowiadających sobie chwilach w cyklu drugim i pierwszym. Natomiast wskaźnik sezonowości w jednej z początkowych chwil i wyznacza się poprzez obliczenie ilorazu i-tej wartości w każdym z N początkowych cykli oraz średniej wartości w danym cyklu, a następnie policzenie średniej tego ilorazu po N pierwszych cyklach. Następnie przewidywana wartość danych w momencie t oraz wartości R, T, S są liczone zgodnie z następującymi wzorami.

$$
\begin{aligned}
R_t &= \alpha \frac{x_t}{S_{t-L}} + (1-\alpha)R_{t-1} + T_{t-1} \\
T_t &= \beta(R_t - R_{t-1}) + (1-\beta)T_{t-1} \\
S_t &= \gamma \frac{x_t}{S_t} + (1-\gamma) * S_{t-L} \\
\bar{x}_{t+m} &= S_t + mT_t S_{t-L+m}
\end{aligned}
$$

Następnie dla przewidywanej wartości w momencie t jest liczony błąd średniokwadratowy na podstawie, którego poprawiane są wartości parametrów α, β, γ. Wytrenowany model powinien dobrać parametry α, β, γ tak, że przewidywane wartości są bliskie rzeczywistym wartościom w serii czasowej.

2.5.2. Zastosowanie metody

Przykład użycia metody Holta-Wintersa można znaleźć w pracy [5]. Naukowcy z Uniwersytetu w Tiumenie użyli tej metody w celu przewidywania 4 parametrów:

- użycia CPU,
- użycia pamięci,
- wejściowego ruchu sieciowego,
- wyjściowego ruchu sieciowego.

dla 2 serwerów, które raportowały te parametry co 10 minut. Z racji, że dla przewidywania serii czasowych ciężko jest użyć klasycznej metody sprawdzianu krzyżowego użyli oni metody sprawdzianu krzyżowego z ruchomym oknem (rozdział 2.4.1) w celu znalezienia optymalnych wartości parametrów α, β, γ w modelu Holta-Wintersa. Rosyjscy badacze potraktowali każdy z parametrów jako oddzielną serię czasową i w ramach testów osiągnęli metodą Holta-Wintersa uśredniony po parametrach błąd MAPE (Mean Absolute Percentage Error) równy 8.7%, co jest znacząco lepszym wynikiem niż błąd na poziomie 11% modelu SARIMA, który został wykorzystany w porównaniu.

Dokładne wartości błędu MAPE predykcji parametrów dla serwerów z różnymi początkowymi wartościami parametrów w metodzie Holta-Wintersa zostały przedstawione na Rysunku 2.6.

Ze względu na wysoką jakość przewidywania serii czasowych metoda Holta-Wintersa została również wykorzystana w pracy [12] w celu wykrywania anomalii w danych ruchu sieciowego. Anomalie zostały sztucznie dodane do zbioru i miały reprezentować hakerski atak *SYN flood* po protokole TCP. Dany ruch sieciowy był uznawany za atak hakerski, jeśli błąd predykcji modelu Holta-Wintersa był większy od ustalonego progu. Próg jednak nie był stały i zmieniał się w

№	Server	Parameter	L	a	β	γ	MAPE
1	1	CPU load	144	1	0	0	14.8
2	1	Memory usage	144	0.9	0	0	2.3
3	1	Network traffic in	144	0.5	0.3	0	3.7
4	1	Network traffic out	144	0.4	0.1	0	11.3
5	2	CPU load	144	0.5	0	0	13.7
6	2	Memory usage	144	0.9	0.1	0	6.4
7	2	Network traffic in	144	0.2	0.1	0	4.4
8	2	Network traffic out	144	0.3	0.1	0	12.7

Rysunek 2.6: Dokładne wartości błędu MAPE dla predykcji różnych parametrów serwerów. (Źródło: [5])

czasie, gdyż za próg została uznana wykładniczo ważona średnia ruchoma opisana równaniem $z_i = \lambda \bar{X}_i + (1 - \lambda) z_{i-1}$, gdzie

- z_i - wartość progu w chwili i,
- λ - ustalony parametr równania,
- $\bar{X}_i$ - błąd predykcji w chwili i.

Takie rozwiązanie umożliwiło modelowi identyfikacje sztucznie dodanych anomalii.

Jednakże metoda potrójnego wygładzania wykładniczego działa tylko na jednowymiarowej serii czasowej, co powodowałoby trudności z wykorzystaniem jej na logach zawierających wiele parametrów.

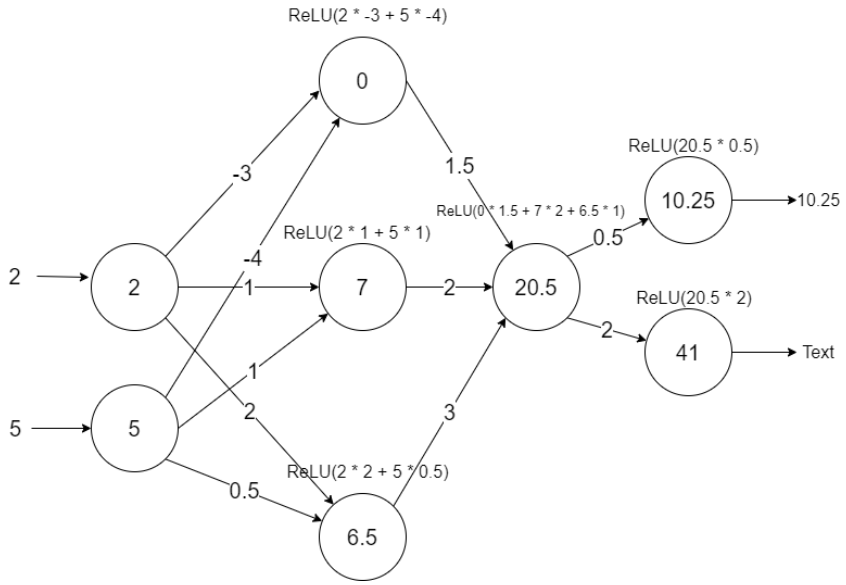
2.6. Sieci neuronowe

Sieci neuronowe to struktury używane w metodach głębokiego uczenia. Ich architektura zainspirowana jest układem nerwowym człowieka, który składa się z neuronów oraz synaps. Sieć neuronową można interpretować jako graf, w którym wierzchołki są odpowiednikiem neuronów, a krawędzie z wagami odpowiadają synapsom. Sieci neuronowe znajdują szerokie zastosowanie w zadaniach klasyfikacji, regresji, przewidywania serii czasowych, znajdowaniu anomalii, rozpoznawaniu obiektów na zdjęciach.

W tej sekcji opisane zostaną typy sieci neuronowych, które zostały zastosowane w wykrywaniu anomalii. Są to między innymi sieci konwolucyjne, rekurencyjne, a także sieci typu AutoEncoder oraz Variational AutoEncoder.

2.6.1. Opis metod

Najprostszym modelem sieci neuronowej jest perceptron wielowarstwowy (Rysunek 2.7). Neurony w sieci zazwyczaj są grupowane w warstwy z czego pierwszą warstwę oraz ostatnią wyróżnia się odpowiednio jako warstwę wejścia i wyjścia. Do neuronów w warstwie wejścia są przekazywane dane, następnie neurony w kolejnej warstwie otrzymują informacje z poprzedniej warstwy. Tą informacją jest wartość będąca sumą iloczynów wartości w neuronie z poprzedniej warstwy oraz wag krawędzi pomiędzy neuronami. Następnie na tę sumę nakładana jest określona funkcja zwana funkcją aktywacji. Za funkcję aktywacji przyjmuje się zazwyczaj tangens hiperboliczny, funkcję sigmoidalną, albo funkcję ReLU. Wynik tej funkcji zostaje zapisany jako wartość przechowywana w neuronie. Wartości przekazywane są do dalszych warstw i proces obliczania wartości dla neuronów w kolejnej warstwie jest powtarzany, aż wartości zostaną przekazane do warstwy wyjścia, która zwraca wartości w swoich neuronach na zewnątrz.

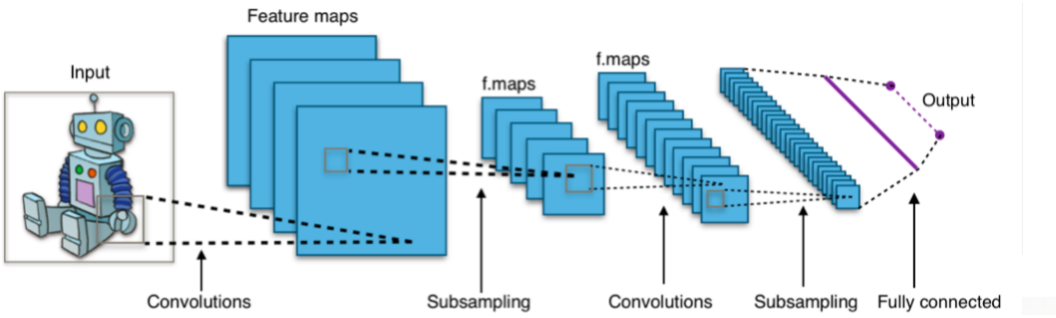


Rysunek 2.7: Przykład obliczeń perceptronu wielowarstwowego z funkcją aktywacji ReLU. (Opracowanie własne)

Trening sieci neuronowych przyjmuje zazwyczaj następujący schemat. Sieć o zdefiniowanej architekturze jest inicjowana, wagi krawędzi są losowane albo wybierane z jakiegoś rozkładu statystycznego. Następnie dla danych wejściowych wykonuje się obliczenia siecią neuronową. Jeśli na wyjściu sieci wartość jest inna niż oczekiwana, to liczony jest błąd (dla zadania regresji może to być błąd średniokwadratowy, a dla zadania klasyfikacji może to być funkcja entropii krzyżowej). Następnie na podstawie błędu poprawiane są wartości wag, zazwyczaj metodą propagacji wstecznej.

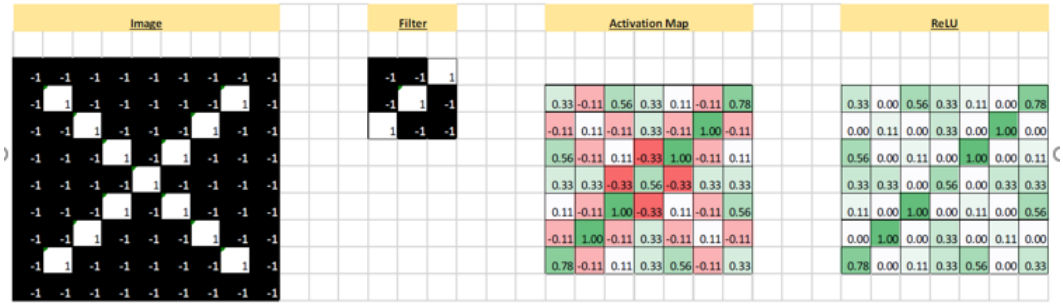
2.6.1.1 Sieci konwolucyjne (CNN)

Jest to typ sieci używany często w zadaniach, w których przetwarzane są obrazy. Potrafią rozwiązywać takie problemy jak rozpoznawanie obiektów na zdjęciach, analiza nagrań, ale także przetwarzanie języka naturalnego. Charakteryzują się tym, że zamiast zwykłych warstw z neuronami zbudowane są z naprzemiennych warstw konwolucyjnych oraz warstw łączących (ang. pooling layer), dopiero ostatnie warstwy sieci przypominają te ze standardowego perceptronu. Przykładowa architektura sieci widoczna jest na Rysunku 2.8. W warstwie konwolucyjnej znaj-



Rysunek 2.8: Architektura sieci CNN[1].

duje się kilka macierzy zwanych filtrami. Filtry te przechodzą po macierzy tworząc mapy aktywacji. Nałożenie filtru w sieci konwolucyjnej wygląda analogicznie jak nałożenie filtru w przetwarzaniu obrazów. Następnie na wartości w komórkach mapy aktywacji nakładana jest funkcja aktywacji. Dzięki temu, sieć może dokonać ekstrakcję prostych cech takich jak np. krawędzie o dowolnej orientacji. Przykład nałożenia filtru w warstwie konwolucyjnej przedstawiony jest na Rysunku 2.9.



Rysunek 2.9: Działanie warstwy konwolucyjnej. (Źródło: Prezentacja: *Convolutional Neural Networks*, Autor: Dominik Lewy)

Warstwa łącząca służy do redukcji rozmiaru przestrzennego macierzy. Polega to na tym, że kilka sąsiednich komórek zastępowanych jest jedną komórką zawierającą najczęściej maksymalną

[1]Źródło: `https://en.wikipedia.org/wiki/Convolutional_neural_network`

wartość spośród nich lub ich średnią. Zmniejszenie rozmiaru danych chroni również sieć przed przeuczeniem. Przykład działania warstwy łączącej został przedstawiony na Rysunku 2.10

Rysunek 2.10: Działanie warstwy łączącej. (Źródło: Prezentacja: *Convolutional Neural Networks*, Autor: Dominik Lewy)

2.6.1.2 Sieci rekurencyjne

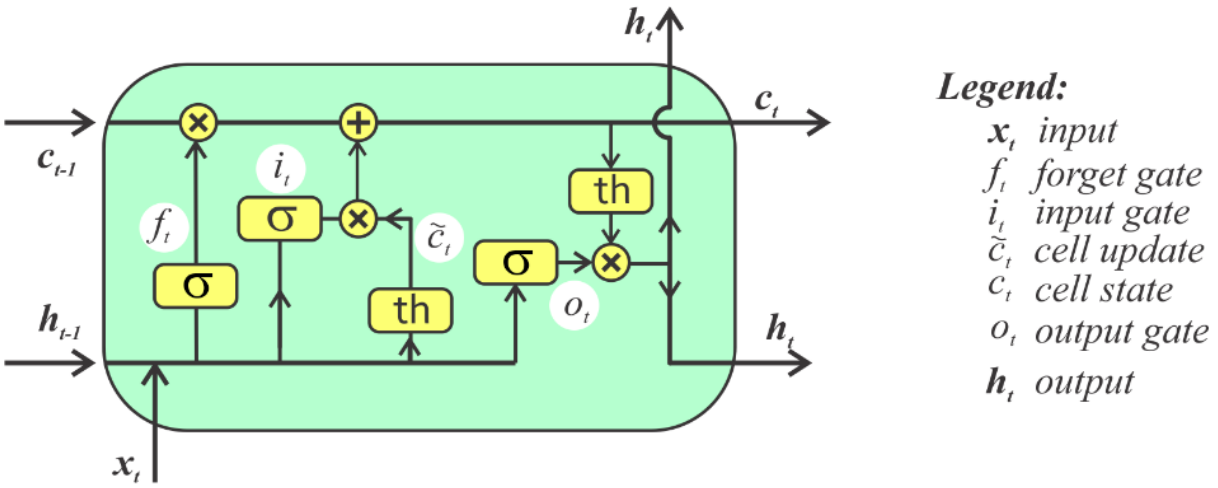
Jest to typ sieci, w których oprócz jednokierunkowego przepływu danych występuje również sprzężenie zwrotne między wejściem a wyjściem. Dzięki temu wyjście sieci nie zależy tylko od stanu wejścia, ale też od wewnętrznego stanu, w którym obecnie znajduje się sieć. Stan wewnętrzny sieci zależy natomiast od poprzednich wartości przekazywanych jako wejście sieci. Wśród podtypów sieci rekurencyjnych można wyróżnić sieci LSTM, sieci Hopfielda, czy sieci BAM.

2.6.1.2.1 Sieci LSTM

Sieci LSTM są jednym z typów rekurencyjnych sieci neuronowych, które są odmianą sieci neuronowych wyróżniającą się tym, że oprócz pojedynczych danych mogą również przetwarzać ich sekwencje. Dzięki temu dobrze radzą sobie z przewidywaniem serii czasowych, rozpoznawaniem mowy. Mechanizm działania LSTM polega na tym, że sieć pamięta kilka ostatnich danych w sekwencji oraz decyduje kiedy pamięć powinna zostać zwolniona. W warstwach sieci LSTM zamiast tradycyjnych neuronów występują tzw. komórki LSTM (Rysunek 2.11).

W komórkach LSTM można wyróżnić następujące elementy:

- bramka zapomnienia (ang. forget gate) – podejmuje decyzje, czy dane obecnie przechowywane w pamięci komórki zostaną zatrzymane, czy informacja zostanie zapomniana,
- bramka wejściowa (ang. input gate) – aktualizuje stan komórki na podstawie obecnego wejścia sieci oraz poprzedniego stanu komórki,
- stan komórki (ang. cell state) – odpowiada za pamiętanie dotychczasowej sekwencji, jest aktualizowany o obecną wartość obliczoną przez bramkę wyjścia,

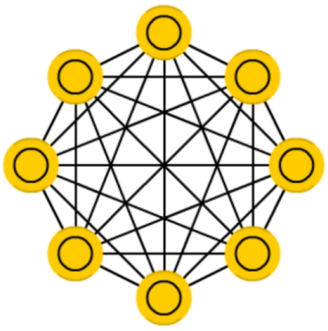


Rysunek 2.11: Budowa komórki LSTM[2].

- bramka wyjścia – jej zadaniem jest obliczenie wartości wyjściowej komórki LSTM na podstawie jej obecnego stanu.

2.6.1.2.2 Sieci Hopfielda

Jest to przykład sieci rekurencyjnej, gdzie wyjścia neuronów są przekazywane z odpowiednimi wagami na wejście każdego z pozostałych neuronów. W sieci Hopfielda każdy neuron jest zarówno neuronem wejściowym jak i wyjściowym. Zadaniem tej sieci jest nauczenie się wzorców wektorowych, do których następnie będą dopasowywane dane wejściowe. Sieć ta jest najczęściej używana do znajdywania wzorców w danych, a jej architektura została przedstawiona na Rysunku 2.12.



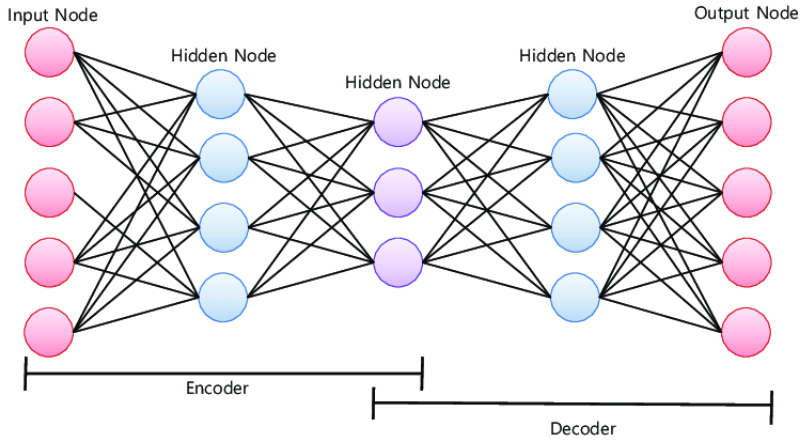
Rysunek 2.12: Przykładowa architektura sieci Hopfielda[3].

[2] https://bhrnjica.net/2019/04/08/in-depth-lstm-implementation-using-cntk-on-net-platform/

[3] https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

2.6.1.3 Sieci typu AutoEncoder

Jest to typ sieci neuronowej, który uczy się skutecznego zakodowania danych. Jego celem jest znalezienie zależności w danych tak, by można było zmniejszyć ich rozmiar, a następnie ze skompresowanego formatu odtworzyć dane początkowe. Cechą wyróżniającą ten typ sieci jest to, że warstwa wejściowa oraz wyjściowa mają zawsze tyle samo neuronów oraz przynajmniej jedna warstwa ukryta musi mieć mniej neuronów niż warstwa wejściowa (by doszło do zmniejszenia rozmiaru danych). Sieci te używane są w celu kompresji danych oraz usuwania szumu z danych. Przykład architektury tej sieci został przedstawiony na Rysunku 2.13



Rysunek 2.13: Przykładowa architektura sieci typu AutoEncoder[4].

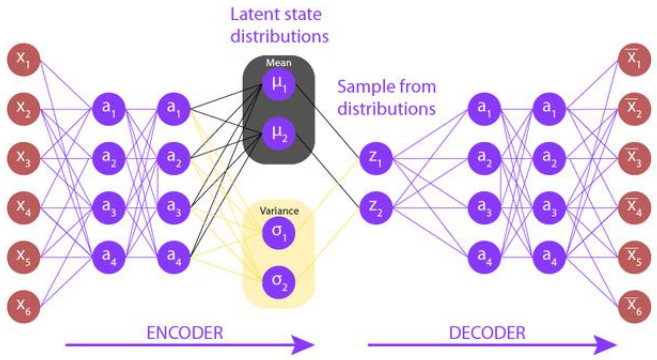
2.6.1.4 Sieci typu Variational AutoEncoder

Jest to modyfikacja sieci typu AutoEncoder. Polega ona na tym, że sieć najpierw wyznacza rozkład gęstości danych wejściowych, a następnie na jego podstawie odtwarza dane wejściowe lub w odpowiedni sposób je modyfikuje. Ten typ sieci może być stosowany w takich samych problemach jak AutoEncoder i dodatkowo może być używany, by tworzyć nowe zbiory danych, np. zamienić męską twarz na twarz kobiety. Architektura tej sieci została przedstawiona na Rysunku 2.14.

2.6.2. Zastosowanie metod

W literaturze można znaleźć wiele przykładów zastosowania metod głębokiego uczenia w celu wykrycia anomalii. Jednym z typów sieci neuronowych, który używany jest przy wykrywaniu anomalii to sieć LSTM. W jednej z prac [6] użyto jej, by wykrywać zaburzenia stanu systemu

[4]Źródło: `https://www.researchgate.net/figure/Structure-of-autoencoder_fig5_322671658`



Rysunek 2.14: Przykładowa architektura sieci typu Variational AutoEncoder[5].

cyberfizycznego (za system cyberfizyczny badacze uznali dowolny izolowany system, który komunikuje się z otoczeniem na podstawie sygnałów wejściowych i wyjściowych). Sieć LSTM w fazie treningu nauczyła się przewidywać stan systemu cyberfizycznego na podstawie jego poprzednich stanów. Następnie badacze założyli, że błędy predykcji przyjmują rozkład normalny. Punkty zostały uznawane za anomalie, jeśli błąd predykcji przekraczał ośmiokrotność odchylenia standardowego. Z takimi parametrami, sieć wykryła 31% rzeczywistych anomalii, co nie brzmi jak dużo. Jednakże aż 95% ze zgłaszanych anomalii było rzeczywistymi anomaliami, dzięki czemu system rzadko podnosił fałszywy alarm.

Sieci LSTM mogą być stosowane w wykrywaniu anomalii w przeróżnych sytuacjach. Autorzy artykułu [4] użyli sieci LSTM na 7 różnych zbiorach danych:

- Vehicular Travel Time – zbiór danych zawierający 2500 odczytów informujących o czasie przejazdu pojazdu przez dany odcinek. Zbiór ten zawiera 8 anomalii.
- Vehicular Speed – zbiór danych zawierający 1128 odczytów pomiaru prędkości pojazdów. W zbiorze występują 3 podsekwencje, które są anomaliami.
- Vehicular Occupancy – 2382 odczytów informujących przez jaki procent czasu w obrębie 30 sekundowego okna sensor wykrywał samochód w polu widzenia. Zbiór zawiera 2 anomalie.
- New York City Taxi Demand - zbiór danych zawierający dane na temat przejazdu taksówek w Nowym Jorku w okresie dwóch miesięcy.
- Bengauluru Taxi Demand – zbiór danych zawierający informacje na temat położenia GPS osób zamawiających taksówkę.
- Electrocardiogram – zbiór danych zawierający 18 000 odczytów elektrokardiogramu zawierający 3 podsekwencje będące anomaliami.

[5]Źródło: `https://www.geeksforgeeks.org/variational-autoencoders/`

- Machine Temperature – zbiór danych zawierający 22 695 odczytów temperatury z maszyny przemysłowej. Zbiór ten zawiera 4 anomalie.

Przewidywana wartość zostawała uznana za anomalie, jeśli błąd predykcji przekraczał określony próg. Naukowcy w ramach badań porównali 3 metody wyznaczania progu anomalii:

- metoda zakładająca, że błędy predykcji mają rozkład Gaussa
- metoda, w której próg anomalii wyznaczono na podstawie teorii wartości ekstremalnych (ang. EVT - Extreme Value Theory)
- metoda oparta na teście Tukey'a ustalająca próg równy $Q_3 + 3 \times (Q_3 - Q_1)$, gdzie Q_1, Q_3 oznaczają odpowiednio górny i dolny kwantyl rozkładu błędów

Najlepszą skuteczność osiągnęła sieć, która wyznaczyła próg anomalii za pomocą metod teorii wartości ekstremalnych, gdyż osiągnęła ona najlepszy wskaźnik F1 na większości zbiorów danych. Wyniki przeprowadzonych testów zostały przedstawione przez naukowców w Tabeli widocznej na Rysunku 2.15.

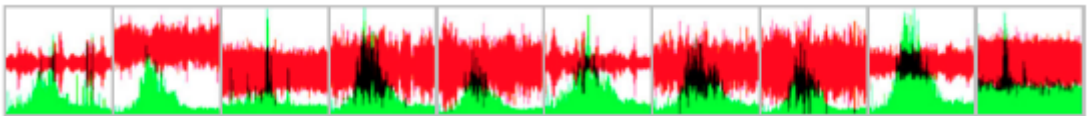
Data Sets		**Normality Assumption**				**Tail Assumption from EVT**				**Tukey's Method**		
		P	R	F1	τ	P	R	F1	q	P	R	F1
Vehicular Travel Time		0.14	0.40	0.21	-20	0.33	0.40	**0.36**	10^{-4}	0.04	0.60	0.07
Vehicular Speed		0.58	1.0	0.73	-18	0.75	0.85	**0.79**	10^{-3}	0.75	0.85	**0.79**
Vehicular Occupancy		1.0	1.0	1.0	-23	1.0	1.0	**1.0**	10^{-5}	0.33	**1.0**	0.5
NYC Taxi Demand	T1	1.0	1.0	**1.0**	-19	1.0	1.0	**1.0**	10^{-5}	1.0	0.14	0.25
	T2	0.2	1.0	0.33	-17	1.0	1.0	**1.0**	10^{-5}	0.07	1.0	0.14
	T3	0.75	1.0	**0.85**	-15	0.75	1.0	**0.85**	10^{-5}	0.5	1.0	0.66
Bengaluru Taxi Demand	T1	1.0	0.4	0.57	-25	1.0	1.0	**1.0**	10^{-4}	0.31	1.0	0.47
	T2	0.33	1.0	**0.5**	-18	0.33	1.0	**0.5**	10^{-4}	0.04	1.0	0.07
	T3	0.6	0.5	0.54	-25	0.57	0.66	**0.61**	10^{-4}	0.15	0.83	0.26
Electrocardiogram		0.50	0.23	0.32	-23	0.50	0.28	0.36	10^{-4}	0.42	0.57	**0.49**
Machine Temperature		0.004	0.50	0.009	-19	0.10	0.50	**0.16**	10^{-4}	0.002	0.50	0.005

Rysunek 2.15: Podsumowanie testów przeprowadzonych przez badaczy. (Źródło: [4])

.

Dodatkowo sukces sieci LSTM w wykrywaniu anomalii na różnych zbiorach danych pokazuje, że jest to metoda, która może być szeroko stosowana w podobnych zagadnieniach.

W znajdowaniu anomalii można również zastosować konwolucyjne sieci neuronowe (CNN). Badacze w pracy [19] opisują zastosowanie właśnie tego typu sieci neuronowej do wykrywania anomalii w serii czasowej pochodzącej z pomiarów medycznych. W badaniach zastosowane zostało nietypowe podejście. Pomiary z urządzenia medycznego podzielono najpierw na przedziały czasowe. Następnie dla każdej serii czasowej, na podstawie wartości pomiarów w czasie, sporządzono wykres częstotliwości występowania wartości pomiaru za pomocą szybkiej transformacji Fouriera. Następnie wykres wartości pomiaru w czasie został pokolorowany na zielono, a wykres częstotliwości został pokolorowany na czerwono, potem oba wykresy zostały zapisane w formie obrazka o wymiarach 100px x 100px. Na koniec oba obrazki zostały nałożone na siebie, czego widok końcowy widać na wykresach przedstawionych przez badaczy (Rysunek 2.16).



Rysunek 2.16: Obrazki otrzymane poprzez nałożenie wykresów wartości pomiaru w czasie oraz wykresu częstotliwości występowania wartości pomiaru. (Źródło: [19])

Z użyciem takiego formatu danych wejściowych naukowcy wytrenowali sieć, która zgłaszała 94.2% anomalii. Dodatkowo tylko 1.1% zgłaszanych anomalii było fałszywym alarmem. Powyższe wyniki mogą napawać optymizmem, jednak nie można pominąć faktu, że sieć konwolucyjna uczy się w trybie nadzorowanym, zatem potrzebuje etykietowanego zbioru danych. W ramach pracy badacze sami przypisali etykiety dla poszczególnych obrazków, jednakże w ogólnym zastosowaniu etykietowanie może być bardzo czasochłonne i kosztowne, zatem powyższa metoda nie może być zastosowana w większości przypadków.

Interesujące podejście można znaleźć w artykule [11] autorstwa Volodymyra Miza i in.. Naukowcy podjęli temat wykrywania anomalii w sieci internetowej oraz sieci społecznościowej. Oprócz wykrywania anomalii starali się zrozumieć ich powód oraz znaleźć zależność jak jedna anomalia wpływa na powstawanie kolejnych. W swojej pracy wykrywali niespodziewane zmiany w ilości wizyt na wybranych podstronach Wikipedii. Zastosowane przez nich podejście składa się z 3 kroków. W pierwszym kroku badacze stworzyli graf, którego wierzchołkami były monitorowane podstrony Wikipedii, a krawędzie między wierzchołkami występowały, gdy odpowiadająca wierzchołkowi podstrona zawierała link do podstrony odpowiadającej drugiemu wierzchołkowi. W drugim kroku dla każdej z obserwowanych podstron wykrywane były anomalie w ilości wizyt danej strony w przeciągu godziny. Wszystkie potencjalne anomalie zostały przekazywane do trzeciego kroku. W trzecim kroku na podstawie grafu połączeń między stronami została utworzona sieć Hopfielda uczona regułą Hebba. Działa ona w taki sposób, że jeśli w miarę krótkim

przedziale czasu na połączonych wierzchołkach wystąpiły anomalie, to waga krawędzi między nimi wzrastała. Dzięki temu, po pewnym etapie nauczania, utworzyły się ścieżki składające się z krawędzi o dużych wagach, które pokazywały propagowanie się anomalii w sieci.

Również sieci neuronowe typu AutoEncoder znajdują zastosowanie w wykrywaniu anomalii. Od klasycznego AutoEncodera skuteczniejsza jest jednak jego odmiana – Variational AutoEncoder (VAE), która lepiej sobie radzi z danymi zawierającymi szum. W jednej z prac [13] VAE został użyty do znajdowaniu anomalii w ruchu sieciowym. Dane na których działali badacze miały bardzo duży rozmiar, w 3-minutowym przedziale czasowym raportowane było 200 000 rekordów NetFlow. Dla każdego z tych przedziałów wyodrębnione zostały 53 cechy i były to między innymi średnia oraz odchylenie standardowe czasu potrzebnego do przesłania pakietów, liczby przesłanych bajtów w pakiecie, czy jaki procent pakietów był przesyłany po kilku wybranych portach (np. WinRPC, Telnet, DNS, SSH, HTTP, FTP, POP3). Cechy zostały oczywiście znormalizowane, a następnie przekazane do sieci neuronowej. Zadaniem modelu było rozpoznanie kilku rodzajów ataków hakerskich takich jak:

1. Atak SYN flood,
2. Skanowanie portów,
3. Atak sieci botów,
4. Przesyłanie spamu poprzez port SMTP.

Model oznaczał dany przedział czasowy jako zawierający anomalie, jeśli błąd średniokwadratowy przy rekonstrukcji cech przekraczał określony próg. W testach sieć VAE poradziła sobie znakomicie, pomimo dużej ilości danych oraz cech wartość AUC pod krzywą ROC wynosiła 0.947. Dodatkowo czas potrzebny dla sieci neuronowej na klasyfikacje przedziału czasowego jest mniejszy od długości przedziału czasowego, co pozwala na nadzorowanie wielkiego ruchu sieciowego w czasie rzeczywistym.

AutoEncodery można również dalej rozszerzać poprzez zastąpienie warstw gęstych w warstwie ukrytej bardziej złożonymi warstwami, czego dokonali badacze w pracy [17]. W sieci typu AutoEncoder w warstwach pośrednich zastosowali warstwy konwolucyjne. Za pomocą takiej architektury modelu próbowali znaleźć anomalie w pomiarach z sensorów monitorujących poziom ścieków w Szwajcarii. Badaczom jednak nie udało się osiągnąć rewelacyjnych wyników, gdyż tylko 35% anomalii zostało wykrytych w danych, dodatkowo system raportował wiele fałszywych alarmów. Jednakże zaletą systemu jest to, że wśród fałszywych pozytywów, część raportowanych anomalii okazała się rzeczywistymi anomaliami, które ze względu na swoją złożoność nie zostały wstępnie wykryte przez eksperta, który etykietował anomalie w zbiorze. Badacze zatem

widzą potencjał w tej architekturze, jednak przyznają, że wymaga ona modyfikacji podejścia do problemu.

2.7. Pozostałe metody

Jeden z ogólnodostępnych modeli do wykrywania anomalii, *Twitter ADVec* [21] jest zawarty w bibliotece *AnomalyDetection*. Jest to pakiet napisany przez programistów Twittera, który umożliwia wykrywanie anomalii w wektorze obserwacji. Biblioteka została zaimplementowana w języku R i używa algorytmu *Seasonal Hybrid ESD* (S-H-ESD). Model ten jest często stosowany przez badaczy w celu porównania jakości ich modeli z innymi dostępnymi narzędziami.

Markus Thill i in. [20] porównują wynaleziony przez nich model z *Twitter ADVec*. Zaprezentowany model *SORAD (Simple Online Regression Anomaly Detector* ma 2 zalety: potrzebuje mniejszej ilości danych w fazie treningu oraz dodatkowo za pomocą współczynnika zapominania, algorytm dostosowuje się do trwałych zmian w szeregu czasowym. Porównanie modeli zostało wykonane na podstawie ich skuteczności na zbiorze S5 opublikowanym przez Yahoo. Jest to zbiór danych z etykietami, który zawiera zarówno rzeczywiste metryki wybranych serwisów Yahoo jak i dane syntetyczne. Oba modele uczyły się w sposób nienadzorowany, etykiety zostały użyte jedynie do wyznaczenia skuteczności modeli. W porównaniu końcowym *SORAD* zdeklasował rywala uzyskując skuteczność o 19-69 punktów procentowych lepszą niż *Twitter ADVec*.

W opisywanym zagadnieniu można również użyć modelu ARIMA [18]. Grupa badaczy z Berlina z sukcesem zastosowała ten model w wersji uczenia Online do danych pochodzących ze środowiska chmurowego. Dane te zostały wzbogacone o sztuczne anomalie takie jak:

- zanieczyszczenie dysku,
- nagłe zmiany w zużyciu procesora,
- nagłe zmiany w zużyciu pamięci,
- tworzenie dużo procesów-dzieci przez pewien proces.

Autorzy pracy oprócz wysokiej skuteczności osiągnęli bardzo mały współczynnik fałszywych pozytywów na poziomie 2%, dzięki czemu system raportujący anomalie zgłaszałby tylko rzeczywiste błędy.

Metody te jednak pracują wyłącznie na danych jednowymiarowych. Traktowanie każdego parametru w logach jako oddzielną serie czasową spowoduje jednak utratę informacji jak dane parametry zależą od siebie.

W zadaniu wykrywania anomalii ostatnio pojawił się również koncept tak zwanych Shapelets. Są one zdefiniowane jako podsekwencja serii czasowej, która jest w pewnym sensie jak najbardziej reprezentatywna dla danej klasy. Można to rozumieć przez to, że w seriach czasowych pochodzących z jednego źródła, jeśli rozpatrywana byłaby seria czasowa od początku do końca, to pewna część wykresu pomiarów ma bardzo podobny kształt. Badacze w pracy [1] wykorzystują ten fakt zakładając, że jeśli wykres pewnych serii czasowych nie zawiera tych kształtów to jest on anomalią. Badaczom udało się tą metodą osiągnąć obiecujące wyniki na seriach czasowych pochodzących z wielu dziedzin, np. medycyny, finansów. Jednak ze względu na to, że każda seria czasowa musi być takiej samej długości oraz mieć określony początek oraz koniec, to metoda ta nie nadaje się w wyszukiwaniu anomalii w logach. Dodatkowo metoda oznaczyłaby całą serie jako anomalie, ale nie potrafiłaby wskazać dokładnego miejsca wystąpienia anomalii.

2.8. Metody wykorzystane na logach raportowanych przez aplikację firmy EMCA

Wcześniej prowadzone były badania w celu wykrywania anomalii na logach pochodzących z systemu firmy EMCA. W tym celu Piotr Janus i in. [9] rozwinęli model OARX (Online AutoRegressive with eXogenous variables) będący rozszerzeniem modelu ARIMA. Model przechowywał informacje na temat powtarzalności serii czasowej w zmiennej egzogenicznej, która była aktualizowana poprzez uśrednianie poprzedniej wartości zmiennej egzogenicznej z wartością obserwowaną w danym momencie. Opracowane podejście również pozwalało na wykrywanie anomalii w danych przesyłanych strumieniowo, co umożliwiało informowanie o występujących błędach w czasie rzeczywistym. Model ten osiągnął niski współczynnik fałszywych pozytywów, dzięki czemu system nie zgłaszał często fałszywych alarmów. Skuteczność OARX została porównana na zbiorach S5 Yahoo z modelami takimi jak Luminol, DSPOT, SR czy DONUT. W tym wypadku OARX okazał się najlepszym modelem (Rysunek 2.17).

Model	A1 Subset			A2 Subset			A3 Subset			A4 Subset		
	F	Pr.	Rec.	F	Pr.	Rec.	F	Pr.	Rec.	F	Pr.	Rec.
SR	0.29	0.29	0.45	0.67	0.58	0.95	0.83	0.76	**0.97**	**0.72**	0.64	**0.95**
Luminol	0.30	0.32	0.46	0.58	0.48	0.85	0.60	0.57	0.76	0.46	0.42	0.67
DSPOT	0.38	**0.49**	0.42	0.72	0.73	0.75	0.42	0.61	0.43	0.23	0.36	0.30
OARX	**0.51**	0.48	**0.58**	**0.95**	**0.95**	**0.97**	**0.86**	**0.96**	0.83	0.66	**0.68**	0.74
Donut	0.35	0.39	0.59	0.47	0.38	0.79	0.33	0.29	0.50	0.29	0.26	0.45

Rysunek 2.17: Porównianie modelu OARX z innymi rozwiązaniami. (Źródło: [9])

Podejście przedstawione w pracy różniło się jednak znacząco od metod wykorzystanych w niniejszej pracy magisterskiej. Dane użyte w pracy [9] były zbierane przez 2 tygodnie z klastra

serwerów, podczas gdy dane użyte w niniejszej pracy pochodzą z jednego serwera oraz zawierają wartości metryk na przestrzeni trzech miesięcy. Dodatkowo logi w pracy Piotra Janusa były tworzone przez inny program, a zatem miały inny format oraz wartości niż dane, które zostały wykorzystane w obecnej pracy. Ponadto metody zastosowane w pracy [9] nie działają na danych wielowymiarowych, w związku z czym w obecnej pracy nie skorzystano z osiągnięć uzyskanych w tamtej pracy, a zamiast tego rozwijano podejście oparte na sieciach neuronowych, które potrafią pracować na danych wielowymiarowych, dzięki czemu zależność między parametrami w logach nie jest zatracona.

2.9. Podsumowanie

Na podstawie przeglądu literatury można powiedzieć, że szeroki wachlarz metod sprawdza się w zadaniu wykrywania anomalii. Część z nich, czyli między innymi model *SORAD*, model *ARIMA*, czy model *OARX* nie sprawdzą się jednak na danych pochodzących z logserwera firmy EMCA ze względu na fakt, że metody te nie działają na danych wielowymiarowych. Co prawda można by rozdzielić zmienne występujące w serii czasowej na oddzielne serie czasowe i następnie na każdej zmiennej zaaplikować oddzielnie jeden z wymienionych modeli. Jednakże takie podejście nie uwzględnia wtedy zależności pomiędzy wieloma zmiennymi, zatem nie bierze pod uwagę stanu całego systemu. Natomiast metody oparte na PCA są bardzo wrażliwe na dobór liczby dominujących składowych głównych oraz nie radzą sobie z wykrywaniem anomalii, gdy te mają bardzo duże wartości, gdyż dochodzi wtedy do zjawiska zanieczyszczenia przestrzeni normalnej. Użycie Shapeletów również nie wchodzi w grę, gdyż metoda ta działa tylko na seriach czasowych, które mają zawsze początek i koniec w tym samym miejscu. Dodatkowo, jako anomalia klasyfikowana jest cała seria czasowa, przez co nie można uzyskać informacji w którym dokładnie miejscu wystąpiła anomalia.

Zatem najlepsze do zastosowania w pracy będą metody głębokiego uczenia. Metody te potrafią pracować na danych wielowymiarowych oraz mogą działać na wycinku serii czasowych, dzięki czemu oprócz zaklasyfikowania anomalii, wskażą również moment jej występowania. W niniejszej pracy postanowiono sprawdzić 2 podejścia. Pierwsze oparte na sieciach predykcyjnych, których zadaniem będzie przewidzenie kolejnych wartości serii czasowych na podstawie poprzednich pomiarów. Drugie podejście wykorzysta natomiast sieci typu AutoEncoder, które będą próbować skompresować kilka kolejnych wartości z logów do pośredniego formatu, a następnie na jego podstawie odtworzyć oryginalną sekwencje. Dla obu podejść pozostanie dodatkowo stworzony model głosujący, złożony z kilku przykładowych architektur, gdyż jest to standardowa metoda używana

w celu osiągnięcia jak najwyższej skuteczności modelu.

3. Dane oraz metodologia

W tym rozdziale zostaną przedstawione zbiory danych, na których wykonywane było wykrywanie anomalii w ramach pracy magisterskiej. Następnie opisane zostaną zastosowane metody, proces pracy nad osiągnięciem jak najlepszego modelu oraz zdefiniowane zostaną miary na podstawie których modele będą oceniane. Dodatkowo w celu ułatwienia czytelnikowi zrozumienia treści pracy, stworzony został słownik pojęć, zawierający objaśnienie terminologii oraz metod używanych w pracy.

3.1. Słownik pojęć

W sekcji tej znajduje się krótkie objaśnienie metod używanych podczas przeprowadzania testów w ramach pracy magisterskiej. Dodatkowo, w sekcji zawarto również definicje pojęć, które używane będą w dalszej części tekstu pracy.

3.1.1. True Positive Rate

TPR mierzy proporcję liczby poprawnie zidentyfikowanych pozytywów do liczby wszystkich pozytywów, zatem zawiera informacje jaki procent anomalii został wykryty przez model. Wskaźnik prawdziwych pozytywów obliczany jest następująco

$$TPR = \frac{TP}{TP + FN},$$

gdzie:

- TP – liczba prawdziwych pozytywów (jest to liczba wykrytych anomalii przez model),
- FN – liczba fałszywych negatywów (jest to liczba anomalii, które nie zostały wykryte przez model).

3.1.2. Precyzja

Precyzją nazywa się iloraz liczby poprawnie zidentyfikowanych pozytywów przez liczbę pozytywów zgłoszonych przez model. W zagadnieniu detekcji anomalii miara ta informuje, jaki

procent anomalii zgłoszonych przez model jest rzeczywistymi anomaliami. Precyzję oblicza się według wzoru:

$$precyzja = \frac{TP}{TP + FP},$$

gdzie:

- TP – liczba prawdziwych pozytywów (jest to liczba wykrytych anomalii przez model),
- FP – liczba fałszywych pozytywów (jest to liczba anomalii, które są zgłaszane przez model, ale nie są anomaliami w rzeczywistości).

3.1.3. Wynik F1

Jest to miara mierząca ogólną skuteczność modelu. Informuje ona o tym jakim ułamek sumy prawdziwych pozytywów, połowy fałszywych pozytywów oraz połowy fałszywych negatywów stanowi liczba prawdziwych pozytywów. Miara ta liczona jest następująco:

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)},$$

gdzie:

- TP – liczba prawdziwych pozytywów (jest to liczba wykrytych anomalii przez model),
- FP – liczba fałszywych pozytywów (jest to liczba anomalii, które są zgłaszane przez model, ale nie są anomaliami w rzeczywistości),
- FN – liczba fałszywych negatywów (jest to liczba anomalii, które nie zostały wykryte przez model).

3.1.4. Uczenie chronologiczne

Jest to proces uczenia modelu, w którym za zbiór treningowy bierze się określoną liczbę obserwacji, które występowały najwcześniej w kolejności chronologicznej. Pozostałe obserwacje, czyli te, które wystąpiły później uznawane są za zbiór testowy. Taki tryb podziału zbioru danych jest często wykorzystywany w pracy nad seriami czasowymi.

3.1.5. Online learning

Metoda uczenia zwana Online learningiem jest procesem, w którym model doucza się na danych przychodzących na bieżąco. Model może uczyć się na pojedynczych instancjach danych lub grupować je w mini-partie (ang. mini-batch). Taki tryb uczenia może zostać wykorzystany do wyuczenia modelu od zera, ale jest również używany jako sposób douczenia wytrenowanego

wcześniej modelu tak, by radził sobie lepiej na nowych danych. W przypadku douczania wytrenowanego modelu, jest to proces szczególnie pomocny, gdy dane zmieniają się z czasem i model zaczyna się gubić.

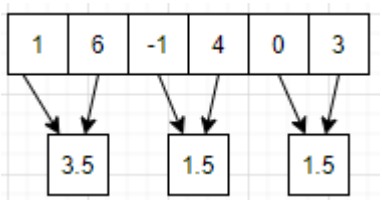
3.1.6. Statystyka zmian

W zadaniu wykrywania anomalii model oznacza dany pomiar jako anomalie, jeśli wykryje w nim niespodziewaną zmianę w porównaniu z poprzednimi pomiarami. Warto jednak zauważyć, że niektóre nagłe zmiany pomiarów wcale nie muszą być czymś niespodziewanym. Przykładowo serwer webowy może codziennie odnotowywać nagły wzrost obciążenia o godzinie 9, gdyż może być to godzina, o której użytkownicy zaczynają logować się do systemu i z niego korzystać. W celu traktowania tych spodziewanych zmian w pomiarach jako coś naturalnego zastosowana została technika nazwana w pracy **Statystyką zmian**. Polega ona na tym, że 24 godzinny okres dnia dzielony jest na przedziały określonej długości, np. 10-minutowe. Następnie błędy odtworzenia/predykcji modelu w zależności od czasu pomiaru przydziela się do odpowiedniego przedziału, a potem do każdego przedziału czasowego przypisywana jest wartość będącą pewnym percentylem wartości błędów w przedziale czasowym pomnożonym przez wybrany przez badacza współczynnik. Wartości te będą używane do potwierdzenie wystąpienia anomalii. Jeśli model zgłosi anomalie występującym w danym pomiarze to dodatkowo dokonywane jest sprawdzenie, czy wartość błędu predykcji/odtworzenia przekracza wartość przypisaną przedziałowi, do którego należy pomiar w Statystyce zmian. Jeśli jest to prawda, to dopiero wtedy uznaje się dany pomiar za anomalię, w przeciwnym przypadku traktuje się anomalię zgłoszoną przez model jako naturalną zmianę w wartościach pomiarów dla tego przedziału czasowego.

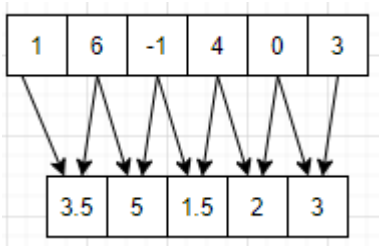
3.1.7. Piecewise Aggregate Approximation

Dane pochodzące z serii czasowych często mogą zawierać dużo szumu przez co kolejne wartości pomiarów mają duże wahania. W celu zredukowania szumu można użyć techniki zwanej **Piecewise Aggregate Approximation** [7], która pozwala wypłaszczyć wykres wartości pomiarów. Polega ona na tym, że na podstawie oryginalnej serii czasowej tworzona jest druga seria czasowa, która jest uśrednieniem oryginalnej serii czasowej. Utworzenie drugiej serii czasowej może przebiegać na 2 sposoby. Pierwszy sposób (Rysunek 3.1) polega na tym, że seria wejściowa jest dzielona na k elementowe rozłączne okna, a następnie każde okno zamieniane jest na jeden pomiar będący średnią wartości pomiarów w jednym oknie. Drugi sposób (Rysunek 3.2) polega na tym, że seria wejściowa jest dzielona na k elementowe nakładające się okna, a potem analogicznie jak w pierwszym przypadku w miejsce okna podmieniany jest jeden pomiar będący średnią arytme-

tyczną pomiarów wchodzących w skład okna. Działanie pierwszego sposobu Piecewise Aggregate Approximation na przykładzie serii czasowej zostało pokazane na Rysunku 3.3.



Rysunek 3.1: Wizualizacja pierwszego sposobu tworzenia serii czasowej w Piecewise Aggregate Approximation (Opracowanie własne)

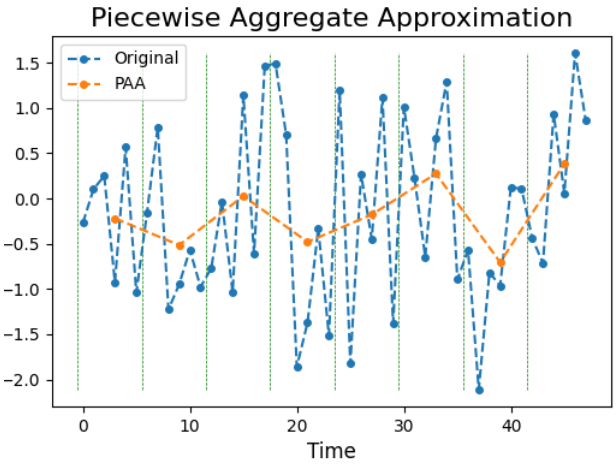


Rysunek 3.2: Wizualizacja drugiego sposobu tworzenia serii czasowej w Piecewise Aggregate Approximation. (Opracowanie własne)

3.1.8. Podstawowy zbiór

W pracy **podstawowym zbiorem** nazywany jest zbiór danych składających się z serii czasowych utworzonych na podstawie zbioru danych pochodzących z logserwera firmy EMCA. Każdy krok czasowy w serii składa się z 5 minutowych przedziałów, a jego parametry to:

- użycie procesora przez procesy systemowe (w procentach),
- użycie procesora przez użytkownika (w procentach),
- nieużywana moc procesora (w procentach),
- używana pamięć RAM (w bajtach),
- wolna pamięć RAM (w bajtach),
- użycie dysku – wartość Configuration (w procentach),
- użycie dysku – wartość Reports (w procentach),
- użycie dysku – wartość Signature (w procentach),
- użycie dysku – wartość Temp (w procentach).



Rysunek 3.3: Wynik działania pierwszego sposobu tworzenia serii czasowej w Piecewise Aggregate Approximation. Na niebiesko zaznaczona jest seria oryginalna, a na pomarańczowo seria utworzona poprzez metodę Piecewise Aggregate Approximation.(Źródło: [7])

3.1.9. Rozszerzony zbiór

Za **rozszerzony zbiór** danych uważa się zbiór danych, który powstał poprzez dodanie do **podstawowego zbioru** takich parametrów jak:

- liczba rekordów typu Event,
- liczba rekordów typu Firewall,
- liczba rekordów typu Health,
- liczba rekordów o priorytecie Information,
- liczba rekordów o priorytecie Notice,
- liczba rekordów o priorytecie Warning,
- liczba rekordów o priorytecie Error,
- liczba wysłanych kilobitów,
- liczba odebranych kilobitów.

3.1.10. Klasyczny podział danych

Jako **klasyczny podział danych** uważa się standardowy proces podziału zbioru danych będących serią czasową na podzbiór treningowy oraz testowy. Polega on na tym, że serię czasową

dzieli się najpierw na nachodzące na siebie okna określonej długości, a następnie te okna ustawiane są w kolejności chronologicznej. Wtedy do podzbioru treningowego, bierze się pierwsze N (gdzie N to pewna ustalona liczba równa np. 80% wszystkich okien czasowych) okien czasowych, a pozostałe okna czasowe stanowią zbiór testowy.

3.1.11. Alternatywny podział danych

Jako **alternatywny zbiór danych** zdefiniowany jest proces podziału zbioru danych będących serią czasową na podzbiór treningowy oraz testowy, który zastosowany został w pracy [15]. Polega on na tym, że seria czasowa dzielona jest na nachodzące na siebie okna określonej długości. Następnie określona liczba okien czasowych jest losowo wybrana jako zbiór treningowy, zaś pozostałe okna czasowe trafiają do zbioru testowego.

3.1.12. Klasyczny sposób oceny

Jest to sposób oceny modeli wykrywania anomalii, w którym:

- liczba prawdziwych pozytywów – jest to liczba anomalii, które zostały poprawnie wykryte przez model,
- liczba fałszywych pozytywów – jest to liczba anomalii zgłoszonych przez model, które nie są rzeczywistymi anomaliami,
- liczba prawdziwych negatywów – jest to liczba pomiarów, w których nie wydarzyła się anomalia oraz dodatkowo model nie zgłosił wtedy wystąpienia anomalii,
- liczba fałszywych negatywów – jest to liczba anomalii, które nie zostały wykryte przez model.

Na podstawie tych metryk następnie potem w standardowy sposób liczone są metryki modelu takiej jak True Positive Rate, precyzja, czy wynik F1.

3.1.13. Alternatywny sposób oceny

W ramach pracy stworzone zostało dedykowane podejście oceniające skuteczność modeli w zadaniu wykrywania anomalii. Ta metoda, nazwana w pracy jako **Alternatywny sposób oceny**, różni się od klasycznej tym, że:

- za prawdziwy pozytyw – uznawany jest pomiar będący rzeczywistą anomalią, wtedy i tylko wtedy gdy model poprawnie zaklasyfikował pomiar jako anomalie lub, gdy model zaklasyfikował jako anomalie przynajmniej jeden z k poprzednich lub k następnych (k –

wybrana liczba całkowita, w testach przyjęte zostało k = 5) pomiarów. Jeśli model ostrzegł przed anomalią kilka chwil wcześniej lub kilka chwil później, to można założyć, że anomalia została poprawnie wykryta. Co więcej zgłoszenie anomalii przed błędem krytycznym jest bardzo pożądane.

- za fałszywy pozytyw – uznawany jest pomiar zgłoszony jako anomalia przez model, wtedy i tylko wtedy gdy k poprzednich lub k kolejnych pomiarów również nie było anomaliami. Jest tak ze względu na to, że niestandardowe zachowanie systemu może występować kilka chwil przed anomalią lub bezpośrednio po restarcie systemu.

- za prawdziwy negatyw – uznawany jest pomiar, w którym nie wydarzyła się anomalia oraz pomiar ten nie został zgłoszony przez model jako potencjalna anomalia

- za fałszywy negatyw – uznawany jest pomiar, w którym wydarzyła się anomalia. Natomiast model nie oznaczył ani dokładnie tego pomiaru jako anomalie, ani żadnego z k poprzednich lub k kolejnych pomiarów.

3.2. Opis danych

Testy w ramach pracy magisterskiej zostały przeprowadzone na 2 zbiorach danych. Pierwszym z nich jest zbiór danych pochodzący z logserwera firmy EMCA. Jest to zbiór, dla którego zadaniem w pracy jest stworzenie jak najskuteczniejszego modelu wykrywającego anomalie, pracę nad tym jednak znacząco utrudnia fakt, że zbiór ten nie zawiera etykietowanych anomalii. Co za tym idzie, podczas procesu prototypowania metod wartych użycia, w celu wykonania oceny danego modelu, to ekspert znający system musiałby przejrzeć, czy w momentach w których model zgłosił anomalie wystąpiły one rzeczywiście. Jest to jednak proces bardzo czasochłonny, a dodatkowo jeśli nie jest znana liczba wszystkich anomalii występujących w zbiorze danych, to nie można policzyć dla modelu takich metryk jak wynik F1, czy True Positive Rate. Jedyną metryką jaką można zmierzyć jest precyzja, jednakże jedynie na jej podstawie nie jest możliwe dokonanie wiarygodnej oceny danego podejścia. Zatem w celu przeprowadzenia rzetelnej oceny modeli potrzebny jest zbiór danych, który jest etykietowany. W tym celu został użyty zbiór danych zawierający logi z maszyny wytwarzającej kartki papieru w fabryce papieru. Został on udostępniony w celach niekomercyjnych przez autorów pracy [15].

3.2.1. Dane pochodzące z logserwera firmy EMCA

Zbiór danych udostępniony przez firmę EMCA składa się z logów zbieranych przez dedykowane rozwiązanie służące do agregacji logów. Dane te zbierane były na przedziale czasowym od 1. grudnia 2020 roku do 31. marca 2021 roku i są logami pochodzącymi z programu Syslog. Syslog jest jednym z narzędzi systemowych dostępnych na systemach uniksowych. W tym przypadku zbierał on rekordy 3 typów:

- Firewall – rekordy informujące o tym, czy zapora sieciowa przepuściła ruch danego pakietu do sieci czy odrzuciła,
- System Health – rekordy zawierające informacje o stanie systemu, np. obciążeniu procesora, zajętości pamięci RAM,
- Event – rekordy informujące o pozostałych wydarzeniach w systemie.

Dodatkowo każdy z rekordów ma przypisany priorytet będący jedną z wartości: Information, Notice, Warning, Error. Dane przed przetworzeniem zajmowały łącznie 107.7 GB i składały się z 86 917 534 rekordów. Rekordy zostały zapisane w formacie CSV. Najważniejszą kolumną w pliku była kolumna message, która zawierała wszystkie informacje na temat rekordu jako słownik klucz-wartość. Na podstawie zawartości pola message, pozostałe kolumny zostały uzupełnianie danymi, a w przypadku, gdy dla danej kolumny w polu message nie było odpowiadającego klucza, to wartość danego parametru była uznawana za pustą wartość. Z racji, że pole message mogło zawierać wiele różnych parametrów, to liczba kolumn w pliku CSV dochodziła nawet do 104. W związku z tym, niemożliwe jest przedstawienie w sposób przejrzysty dla czytelnika przykładowego rekordu w pliku CSV. Jednakże, tak jak było to już wspomniane wartości kolumn były uzupełniane na podstawie wartości pola message, zatem wystarczy pokazać zawartość tego właśnie pola, by zobaczyć całą zawartość danego rekordu.

Zawartość pola message dla przykładowego rekordu typu System Health została przedstawiona na Rysunku 3.4. Na podstawie wartości System Health dla klucza log_type wiadome jest, że jest to rekord informujący o stanie systemu. Na podstawie wartości Memory dla klucza log_component można określić, że rekord będzie zawierał informacje na temat pamięci RAM. W szczególności rekord ten informuje, że całkowica pamięć (klucz total_memory) wynosi 4 145 111 040 bajtów (jednostka odczytana z klucza unit), z czego 1 530 716 160 bajtów jest obecnie używane (na podstawie wartośći pola used).

Natomiast zawartość pola message dla przykładowego rekordu typu Firewall została zobrazowana na rysunku 3.5. Wartość Firewall oraz Firewall rule w kluczach log_type oraz

```
<30>device="SFW"
date=2021-02-13
time=00:15:38
timezone="CET"
device_name="SFV4C6"
device_id=C010012WDG4HY9D
log_id=127726618031
log_type="System Health"
log_component="Memory"
log_subtype="Usage"
priority=Information
unit=byte
total_memory=4145111040
free=2614394880
used=1530716160
```

Rysunek 3.4: Zawartość kolumny message dla przykładowego rekordu typu System Health. (Opracowanie własne)

log_component informują, że dany rekord zawiera szczegóły dotyczące decyzji podjętej na podstawie zasady Firewall. Na podstawie wartości Allow w kluczu status można dowiedzieć się, że dany pakiet został przepuszczony przez zaporę sieciową. Pakiet pochodził z komputera o IP 95.40.123.161 (klucz src_ip) i został przesłany na adres IP 31.179.270.196 (klucz dst_ip). Sam rekord nie niesie ważnej informacja jeśli chodzi o wykrywanie anomalii, jednakże liczność rekordów typu Firewall w danym odstępie czasu ma znaczenie. Ich wzmożona liczba może świadczyć o ataku DDoS (Distributed Denial of Service).

```
<30>device="SFW" date=2021-02-13 time=13:05:51
timezone="CET" device_name="SFV4C6" device_id=C010012WDG4HY9D
log_id=010101600001 log_type="Firewall" log_component="Firewall Rule"
log_subtype="Allowed" status="Allow" priority=Information
duration=0 fw_rule_id=39 policy_type=3
user_name="" user_gp="" iap=0
ips_policy_id=0 appfilter_policy_id=0 application=""
application_risk=0 application_technology="" application_category=""
in_interface="PortC" out_interface="PortA" src_mac=00:00:00:00:00:00
src_ip=95.40.123.161 src_country_code=POL dst_ip=31.179.250.196
dst_country_code=POL protocol="TCP" src_port=36670
dst_port=995 sent_pkts=0  recv_pkts=0
sent_bytes=0 recv_bytes=0 tran_src_ip=
tran_src_port=0 tran_dst_ip=10.4.4.1 tran_dst_port=0
srczonetype="WAN" srczone="WAN" dstzonetype="LOCAL"
dstzone="LOCAL" dir_disp="" connevent="Start"
connid="464289792" vconnid="" hb_health="No Heartbeat"
message="" appresolvedby="Signature" app_is_cloud=0
```

Rysunek 3.5: Zawartość kolumny message dla przykładowego rekordu typu Firewall. (Opracowanie własne)

W obecnym zadaniu najważniejsze są jednak informacje, które można wydobyć z rekordów typu System Health, gdyż na ich podstawie można ocenić stan systemu. Rekordy tego typu są raportowane w 5 minutowych odstępach i na ich podstawie stworzona została seria czasowa nazwana w pracy **Podstawowym zbiorem**. Seria ta składa się z 5 minutowych przedziałów, w której każdy krok czasowy zawierał takie informacje jak:

- użycie procesora przez procesy systemowe (w procentach),
- użycie procesora przez użytkownika (w procentach),
- nieużywana moc procesora (w procentach),
- używana pamięć RAM (w bajtach),
- wolna pamięć RAM (w bajtach),
- użycie dysku – wartość Configuration (w procentach),
- użycie dysku – wartość Reports (w procentach),
- użycie dysku – wartość Signature (w procentach),
- użycie dysku – wartość Temp (w procentach).

Jednakże logi pochodzące z serwera zawierały czasem braki. W przypadku, gdy brakowało rekordów w obrębie jednego kroku czasowego, to wymienione wyżej wartości zostały zinterpolowane jako średnia wartości parametrów z poprzedniego oraz następnego kroku czasowego. W przypadku, gdy przerwa w rekordach wynosiła dłużej niż jeden krok czasowy (więcej niż 5 minut), to dana seria czasowa była uznawana za zakończoną, a od momentu, gdy rekordy zaczęły ponownie być zbierane przez logserwer rozpoczynano kolejną serię czasową.

Warto jednak zwrócić uwagę, że rekordy typu System Health stanowią jedynie ułamek wszystkich rekordów w logach. Pomimo tego, że rekordy typu Event oraz Firewall nie niosą ważnej dla modelu informacji, to warto byłoby je w jakiś sposób uwzględnić jako parametr dla danego kroku czasowego. W tym celu stworzony został **Rozszerzony zbiór**, który powstał poprzez wzbogacenie **Podstawowego zbioru** o parametry takie jak:

- liczba rekordów typu Event w 5 minutowym oknie,
- liczba rekordów typu Firewall w 5 minutowym oknie,
- liczba rekordów typu Health w 5 minutowym oknie,
- liczba rekordów o priorytecie Information w 5 minutowym oknie,

- liczba rekordów o priorytecie Notice w 5 minutowym oknie,
- liczba rekordów o priorytecie Warning w 5 minutowym oknie,
- liczba rekordów o priorytecie Error w 5 minutowym oknie,
- liczba wysłanych kilobitów w 5 minutowym oknie,
- liczba odebranych kilobitów w 5 minutowym oknie.

Jest to cenna informacja, gdyż przykładowo nagły wzrost liczby rekordów typu Firewall może świadczyć o próbie ataku DDoS, a niespodziewanie większa liczba rekordów typu Warning może sygnalizować pewnego rodzaju błąd występujący w systemie.

Obie serie czasowe, czyli **Podstawowy zbiór** oraz **Rozszerzony zbiór** zawierały 26 588 rekordów, co przekładało się na zajętość pamięciową równą 3.4MB. Przykładowe rekordy z Rozszerzonego zbioru zostały przedstawione w Tabeli 3.1 (takie same rekordy tylko bez opisanych wyżej pól znajdują się również w zbiorze podstawowym). Tabela została podzielona na 2 części ze względu na dużą liczbę kolumn dla danego rekordu. Analizując pierwszy rekord dowiemy się, że opisuje on 5 minutowe okno czasowe zaczynające się 1. grudnia 2020 roku o godzinie 00:00:00 (kolumna timestamp) i dodatkowo w tym przedziale czasowym:

- nie wystąpiły żadne rekordu typu Event (kolumna event_count),
- wystąpiło 1289 rekordów typu Firewall (kolumna firewall_count),
- wystąpiło 10 rekordów typu System Health (kolumna health_count),
- wystąpiło 215 rekordów o priorytecie Notice (kolumna notice_count),
- wystąpiło 1084 rekordów o priorytecie Info (kolumna info_count),
- wystąpiło 0 rekordów o priorytecie Warning oraz Error (kolumny warning_count oraz error_count),
- średnie zużycie pamięci RAM wynosiło 1 987 613 000 bajtów (kolumna mem_used), a wolną pamięć RAM wynosiła 2 157 568 000 bajtów (kolumna mem_free),
- wysłane zostało 5996.90 kilobitów (kolumna transmitted_kbits) oraz odebrane zostało 6195.98 kilobitów (kolumna received_kbits),
- użycie dysku przyjęło wartości Configuration 18% (kolumna disk_conf),Temp 3% (kolumna disk_temp), Reports 21% (kolumna disk_reports), Signature 22% (kolumna disk_signature),

- procesy użytkownika konsumowały 3.68% mocy procesora (kolumna cpu_user), 2.07% mocy procesora obsługiwało procesy systemowe (kolumna cpu_system), a pozostałe 94.25% mocy procesora było nieużywane (kolumna cpu_idle)
- wszystkie uwzględnione metryki były zaraportowane (kolumna contains_na) – informacja z tej kolumny nie była przekazywana do sieci

timestamp	event_count	firewall_count	health_count	notice_count	info_count	warning_count	error_count	mem_used	mem_free
2020-12-01 00:00:00	0	1289	10	215	1084	0	0	1.987613e+09	2.157568e+09
2020-12-01 00:05:00	7	1144	10	223	937	1	0	1.991729e+09	2.153452e+09
2020-12-01 00:10:00	2	1187	10	217	990	0	0	1.991115e+09	2.154066e+09
2020-12-01 00:15:00	8	1087	10	236	869	0	0	1.989059e+09	2.156122e+09
2020-12-01 00:20:00	0	1120	10	218	912	0	0	1.972785e+09	2.172396e+09

transmitted_kbits	received_kbits	disk_conf	disk_temp	disk_reports	disk_signature	cpu_user	cpu_system	cpu_idle	contains_na
5996.90	6195.98	18.0	3.0	21.0	22.0	3.39	2.50	94.11	0
2827.39	2986.55	18.0	3.0	21.0	22.0	3.07	1.85	95.07	0
1536.67	1737.29	18.0	3.0	21.0	22.0	3.10	1.91	94.99	0
1379.36	1560.16	18.0	3.0	21.0	22.0	3.15	1.82	95.02	0
721.56	787.31	18.0	3.0	21.0	22.0	3.68	2.07	94.25	0

Tablica 3.1: Przykładowe rekordy z zbioru rozszerzonego.

3.2.2. Dane pochodzące z fabryki papieru

Tak jak wcześniej już wspomniano, w zbiorze danych firmy EMCA anomalie nie są oznaczone. Tak więc naturalnym jest zatem, że sieci muszą się uczyć w trybie nienadzorowanym. Jednakże w celu oceny modeli potrzebny jest etykietowany zbiór, na którym można by było zmierzyć takie metryki jak accuracy, recall, precision, czy F1-score danego modelu. W tym celu wykorzystano zbiór danych użyty w pracy [15]. Zawiera on pomiary z urządzenia wytwarzającego kartki papieru w fabryce papieru. Zbiór ten składa się z 18 398 pomiarów wykonywanych co 2 minuty, a w kolumnach każdego rekordu można znaleźć:

- informacje, czy w danym momencie doszło do awarii maszyny (czy wydarzyła się anomalia); w zbiorze jest 124 anomalii
- zmienną kategoryczną – 2 kolumny
- zmienną numeryczną – 59 kolumn

Użycie tego zbioru w testach zagwarantowało możliwość oceny skuteczności metod stosowanych w pracy dyplomowej. Dodatkowo autorzy pracy [15] zasugerowali sposób podziału danych na okna czasowe zawierające po 5 pomiarów oraz odrzucenie kolumn ze zmienną kategoryczną w procesie uczenia. Opublikowali oni również wyniki ich najlepszego modelu, które wynoszą:

- precyzja – 0.071,
- wynik F1 – 0.114,
- False Positive Rate (FPR) – 0.026.

3.3. Metodologia

Po analizie danych, które użyte zostaną w badaniach przeprowadzonych w ramach pracy magisterskiej, opisane zostaną modele użyte podczas testów oraz szczegóły ich implementacji. Następnie zdefiniowane zostaną miary skuteczności modeli na podstawie których ocenione zostało każde podejście. Na koniec opisany zostanie proces przeprowadzonych badań.

3.3.1. Modele wykorzystane w badaniach

Tak jak zostało to już wspomniane w sekcji 2.9 ze względu na fakt, że serie czasowe utworzone z logów firmy EMCA są wielowymiarowe, uniemożliwia to wygodne zastosowanie standardowych metod uczenia maszynowego ze względu na to, że nie są przystosowane do pracy z danymi wielowymiarowymi. Dlatego w pracy zdecydowano się przeprowadzić testy używając sieci neuronowych. Użyte sieci można podzielić na 2 kategorie:

- Sieci typu AutoEncoder – sieci, które sprowadzają dane z kilku pomiarów do skompresowanej postaci, a następnie na jej podstawie odwtarzają dane wejściowe,
- Sieci przewidujące – sieci, które na podstawie pomiarów dla kilku poprzednich kroków czasowych starają się przewidzieć wartość kolejnego pomiaru.

Dodatkowo w obrębie jednej kategorii sieci zostaną połączone w model głosujący. Jest to często stosowane podejście, które pozwala osiągać model o najlepszej skuteczności. Zatem w ramach testów sprawdzone zostały następujące architektury:

1. Sieci typu AutoEncoder:
    - AutoEncoder CNN (sekcja 6.1, Rysunek 6.1a),
    - AutoEncoder LSTM (sekcja 6.1, Rysunek 6.1b),
    - 2 sieci CNN LSTM (sekcja 6.1, Rysunki 6.2a, 6.2b),
    - model głosujący z powyższych sieci.
2. Sieci przewidujące:

- CNN (sekcja 6.1, Rysunek 6.3a),
- LSTM (sekcja 6.1, Rysunek 6.3b),
- CNN LSTM (sekcja 6.1, Rysunek 6.4),
- model głosujący z powyższych sieci.

Rysunki architektur można znaleźć w dodatku w rozdziale 6.1. Sieci były trenowane na platformie Google Colab ze względu na dostęp do wirtualnych maszyn z możliwością akceleracji GPU, co wysoce przyspiesza działanie sieci neuronowych. Wszystkie skrypty użyte podczas testów zostały napisane w języku Python. Sieci neuronowe zostały zaimplementowane z użyciem biblioteki *Keras* oraz pakietu *Keras Self-Attention*, który zawiera gotowe do użycia warstwy self-attention. Do przetwarzania danych zostały wykorzystane pakiety *Pandas* oraz *NumPy*. Poza tym, w celu normalizacji danych oraz podziału zbioru danych na zbiór treningowy i testowy użyty został pakiet *sklearn*. Natomiast do sporządzania wykresów została wykorzystana biblioteka *matplotlib*.

3.3.2. Miary skuteczności modeli

Żeby umożliwić porównanie modeli w celu określenia, który z nich jest najlepszy trzeba zdefiniować najpierw miary skuteczności modeli. Zagadnienie wyszukiwania anomalii jest jednak dość specyficznym problemem. Zadanie to można sprowadzić do problemu klasyfikacji binarnej odpowiadającą na pytanie, czy w danej chwili zaszła anomalia. Wtedy dla każdego kroku czasowego można przypisać wartość prawdziwą, jeśli w danym momencie wystąpiła anomalia, albo wartość fałszywą, jeśli w danym momencie system działał w standardowych warunkach. Ze względu na fakt, że anomalie w systemach występują bardzo rzadko, to występuje tutaj problem niezbalansowania klas. Z tego powodu przykładowo jedna z najczęściej używana metryka w ocenie modeli, czyli accuracy jest tutaj całkowicie niemiarodajna. Skoro anomalie pokrywają mniej niż 1% kroków czasowych, to model, który zawsze by zwracał wartość fałszywą osiągnął by accuracy powyżej 99% pomimo tego, że jest on totalnie niepoprawnym modelem. Zatem, żeby ocenić model w pracy wybrano metody oparte na liczbie prawdziwych pozytywów.

Pierwszą miarą, która jest brana pod uwagę przy oceny modeli jest TPR (ang. True Positive Rate). Mówi ona o tym jaki procent anomalii jest wykrywany przez model. Jest to ważna metryka, gdyż w rozwiązaniach produkcyjnych ważne jest, żeby jak największa liczba anomalii była wykrywana, gdyż niewykrycie anomalii może potencjalnie spowodować problemy w dostarczaniu przez firmę danych usług, co może spowodować przejście klientów do konkurencji.

Drugą miarą względem której oceniane są modele, to precyzja (ang. precision). Informuje ona o tym, jaki procent anomalii zgłaszanych przez model to rzeczywiste anomalie. Naturalne jest, że pożądane jest, żeby wartość tej metryki osiągała jak największą wartość. Jeśli model podnosiłby

często tak zwany fałszywy alarm, to specjalista weryfikujący wystąpienie anomalii nie wyrabiałby się z sprawdzeniem każdej potencjalnej anomalii.

Trzecią i zarazem ostatnią metryką oceny modeli jest wynik F1. Na jej podstawie można określić jaki ułamek stanowią poprawnie wykrywane anomalie w stosunku do błędnych decyzji podejmowanych przez model, czyli liczby zgłoszonych fałszywych pozytywów oraz liczby anomalii, które nie zostały zgłoszone przez model. Miara ta pozwala ocenić jak w ogólności radzi sobie nasz model przy założeniu, że nieważna jest sytuacja w której model zaklasyfikowuje poprawnie wartość fałszywą. Taka sytuacja ma miejsce w wykrywaniu anomalii, gdyż okres w którym anomalie nie występują nie ma większego znaczenia na działanie systemu.

3.3.3. Proces przeprowadzonych badań

W ramach przeprowadzonych badań dane zostały podzielone na okna składające się z 5 pomiarów, tak jak zaproponowano w pracy [15]. Zadaniem sieci typu AutoEncoder była kompresja sekwencji wejściowej, a następnie odtworzenie jej. Po ukończonym procesie trenowania, sieć odwzorowywała wszystkie okna ze zbioru treningowego, a następnie dla każdego okna liczony był błąd odwzorowania jako błąd średniokwadratowy. Następnie na podstawie wartości błędów była ustalana wartość graniczna (ang. threshold), równa pewnemu percentylowi (w badaniach był to 98 percentyl) wartości błędów odwzorowania. Jeśli błąd odwzorowania przekraczał wartość graniczną, to można było uznać, że dane okno czasowe zawierało pomiar będący anomalią. Jednakże nie wiadomo, który z pomiarów w oknie czasowym jest anomalią. Można go jednak znaleźć, gdyż jeśli dany pomiar jest anomalią, to wszystkie okna czasowe zawierające go również będą anomalią. Zatem jeśli N (N = długość okna czasowego) kolejnych okien czasowych jest anomalią, to pomiar, który występuje w każdym z tych okien jest szukaną anomalią.

Natomiast dla sieci przewidujących dane zostały podzielone na okna składające się z 6 pomiarów. Zadaniem tych sieci była predykcja wartości 6-go pomiaru na podstawie wartości pierwszych 5-ciu pomiarów w oknie czasowym. Po skończonym procesie uczenia sieci, podobnie jak dla sieci typu AutoEncoder wyznaczona została wartość graniczna równa 98 percentylowi błędów predykcji wartości pomiarów na zbiorze treningowym. Jeśli błąd predykcji przekraczał wartość graniczną, to 6-ty (przewidywany) pomiar w oknie czasowym uznawany był za anomalię.

Podczas testów pojawił się pewien problem. W standardowym procesie oceny modelów dla okien czasowych zbiór danych jest dzielony chronologicznie tak, że pierwsze 80% pomiarów jest przydzielane do zbioru treningowego, natomiast ostatnie 20% pomiarów przydzielane jest do zbioru testowego. Zastosowane sieci bardzo dobrze sobie radziły z odwzorowaniem oraz predykcją wartości na zbiorze treningowym, lecz ich skuteczność drastycznie spadała na zbiorze testowym.

Zazwyczaj świadczy to o przetrenowaniu modelu, lecz zmniejszanie parametru modeli w żadnym stopniu nie wpłynęło na to zachowanie. Zatem można wysunąć tezę, że z racji, że pomiary zostały zebrane jedynie na przestrzeni 25 dni, to pierwsze 80% danych prawdopodobnie nie są wystarczająco reprezentatywne. W tym wypadku oprócz standardowego procesu oceny modeli, modele zostały ocenione w dedykowanym trybie zwanym w pracy **Alternatywnym podziałem danych**. Sieci typu AutoEncoder otrzymały jako zbiór treningowy 80% losowo wybranych okien czasowych, a następnie ich celem było odtworzenie wszystkich okien czasowych w zbiorze danych. Ocena modelu jest przeprowadzana na całym zbiorze danych, gdyż jak opisano wcześniej, w celu określenia pomiaru jako anomalii potrzebne jest kilka kolejnych okien czasowych. Natomiast ocena na danych, które zostały przekazane podczas treningu nie są problemem z racji na to, że zadaniem tego typu sieci jest kompresja oraz odtworzenie danych wejściowych ze spakowanego formatu. Sieci przewidujące również otrzymały jako zbiór treningowy 80% losowo wybranych okien czasowych, jednak ich ocena została przeprowadzona na podstawie predykcji kolejnych wartości dla pozostałych 20% okien czasowych.

Również ocena modeli przebiegała w sposób niestandardowy, ponieważ w klasycznym podejściu liczba prawdziwych pozytywów jest równa liczbie rzeczywistych anomalii zgłoszonych przez model, a liczba fałszywych pozytywów to liczba anomalii zgłoszonych przez model, które w rzeczywistości nie są anomaliami. Nie jest to jednak idealna miara w przypadku wykrywania anomalii, gdyż w zbiorze danych anomalią jest oznaczony pomiar, po którym wystąpiła awaria systemu, jednakże niestandardowe zachowanie systemu można zaobserwować kilka pomiarów wcześniej. Dodatkowo pomiary po restarcie, gdy system rozpoczyna działanie również mogą odchodzić od stanu normalnego. Właśnie dlatego, w ramach pracy magisterskiej, stworzone zostało dedykowane podejście określające liczbę prawdziwych oraz fałszywych pozytywów (ang. true and false positives) dostosowane do problemu wykrywania anomalii. Ta metoda, nazwana w pracy jako **Alternatywny sposób oceny**, różni się od klasycznej, gdyż:

- za prawdziwy pozytyw – uznawany jest pomiar będący rzeczywistą anomalią, wtedy i tylko wtedy gdy model poprawnie zaklasyfikował pomiar jako anomalie lub, gdy model zaklasyfikował jako anomalie przynajmniej jeden z k poprzednich lub k następnych (k – wybrana liczba całkowita, w testach przyjęte zostało k = 5) pomiarów. Jeśli model ostrzegł nas przed anomalią kilka chwil wcześniej lub kilka chwil później, to można założyć, że anomalia została poprawnie wykryta. Co więcej zgłoszenie anomalii przed błędem krytycznym jest bardzo pożądane.

- za fałszywy pozytyw – uznawany jest pomiar zgłoszony jako anomalia przez model, wtedy i tylko wtedy gdy k poprzednich lub k następnych pomiarów również nie było anomaliami.

Jest tak ze względu na to, że niestandardowe zachowanie systemu może występować kilka chwil przed anomalią lub bezpośrednio po restarcie systemu.

- za prawdziwy negatyw – uznawany jest pomiar, w którym nie wydarzyła się anomalia oraz pomiar ten nie został zgłoszony przez model jako potencjalna anomalia.

- za fałszywy negatyw – uznawany jest pomiar, w którym wydarzyła się anomalia. Natomiast model nie oznaczył ani dokładnie tego pomiaru jako anomalie, ani żadnego z k poprzednich lub k kolejnych pomiarów.

Ten sposób może jednak wydawać się dla niektórych kontrowersyjny, dlatego w pracy modele były oceniane zarówno klasycznym jak i alternatywnym sposobem oceny. Jednak przy wyborze najlepszego modelu kierowano się wynikami modelu w alternatywnym sposobie oceny.

Testy zostały przeprowadzone najpierw na danych z fabryki z racji, że zbiór ten jest etykietowany. Sam proces rozwoju modeli wykrywania anomalii przebiegał w trybie iteracyjnym. Na początku wykorzystano modele opisane w sekcji 3.3.1. Przyniosły one satysfakcjonujące rezultaty, jednakże celem pracy było uzyskanie jak najskuteczniejszego modelu zatem pierwsza modyfikacja podejścia polegała na modyfikacji architektur zastępując warstwy dropoutu na warstwy Gaussian dropout. Dodatkowo architektury zostały wzbogacone o warstwę self attention. Dzięki temu skuteczność modeli uległa lekkiej poprawie. Kolejna próba poprawy podejścia polegała na próbie usunięcia trendu z zbioru danych, gdyż okazało się, że wartości w zbiorze danych rosły z czasem. Polegało na to na tym, że wartość metryk w danym kroku czasowym została zastąpiona różnicą dwóch kolejnych wartości metryk. Podejście to jednak spowodowało spadek skuteczności modeli zatem zostało ono porzucone. W następnym kroku skupiono się na próbie poprawy skuteczności modeli podczas gdy zbiór danych jest dzielony na podzbiór treningowy oraz testowy w kolejności chronologicznej. W tym celu zastosowano metodę Online learningu, w której modele po wykonaniu klasyfikacji na pewnej części nowych danych dokonywały aktualizacji wag sieci na podstawie tych właśnie danych. Dzięki temu modele poprawiły swoją skuteczność w sytuacji, gdy zbiór treningowy nie składa się z losowo wybranych okien czasowych. Następnie starania zostały skierowane na poprawę precyzji modeli, gdyż sytuacja w której model zgłaszałby wiele fałszywych pozytywów jest bardzo niepożądana, gdyż wymagałaby ona częstej interwencji eksperta, który musiałby sprawdzić wszystkie potencjalne anomalie. W tym celu wykorzystano statystykę zmian, która łaskawiej traktuje błędy modeli w okresach dnia, w których model słabo sobie radzi z odtworzeniem danych w przypadku AutoEncoderów czy przewidzeniem kolejnej wartości w przypadku sieci predykcyjnych. Po zastosowaniu tej metody, precyzja modeli rzeczywiście się zwiększyła, a w celu dalszej jej poprawy użyto ostatnią metodę zastosowaną w pracy,

czyli Piecewise Aggregate Approximation, która polegała na utworzeniu nowej serii czasowej. W nowej serii, każdy pomiar jest średnią 2 przyległych pomiarów i na takiej właśnie serii pracowały modele. W tym przypadku wyniki były jednak niejednoznaczne, część modeli uległa poprawie, lecz niektóre modele odnotowały lekki spadek precyzji. Gdy już otrzymane zostało najefektywniejsze podejście w celu wykrywania anomalii, zostało ono wykorzystane na zbiorze firmy EMCA. Zgłoszone przez modele głosujące anomalie zostały następnie zweryfikowane przez specjalistę, co pozwoliło na obliczenie precyzji modeli na danych zebranych z logsewera. Ze względu na brak informacji o liczbie oraz występowaniu wszystkich anomalii, nie można jednak wyznaczyć takich miar modeli jak TPR oraz wynik F1.

4. Wyniki przeprowadzonych badań

W rozdziale tym przedstawione są wyniki modeli osiągnięte podczas rozwoju metody podejścia do problemu na podstawie danych z pracy [15]. W dalszej sekcji 4.2 omówiona została precyzja jaką osiągnęły modele głosujące na logach pochodzących z logsewera firmy EMCA.

4.1. Zastosowanie wybranych metod wykrywania anomalii do danych pochodzących z fabryki papieru

Na początku przedstawione zostaną wyniki osiągnięte na danych z fabryki papieru z racji, że na tych danych rozwijane były metody. Dzięki temu, że zbiór był etykietowany pozwolił na dokładną ocenę modeli. Ze względu na fakt, że alternatywny tryb oceny modelu jest najbardziej odpowiedni do oceny modeli w zadaniu wykrywania anomalii, to przy porównaniu modeli skupiono się głównie na metrykach modeli obliczonych przez zastosowanie alternatywnego trybu oceny modeli. Tabele zawierające oceny modeli w klasycznym sposobie oceny pełnią raczej rolę ciekawostki oraz stanowią przyszłym badaczom punkt odniesienia, jeśli nie chcieliby oni używać alternatywnego sposobu oceny.

4.1.1. Ocena modeli podstawowych

Na początku, ocenie poddane zostały modele przedstawione w sekcji 3.3.1. W celu oceny skuteczności modeli obliczono metryki takie jak współczynnik prawdziwych pozytywów (ang. TPR – True Positive Rate), precyzja (ang. precision) oraz wynik F1. Na podstawie wyników zawartych w Tabelach 4.1, 4.2, 4.3, 4.4 widać, że sieci przewidujące radzą sobie lepiej od AutoEncoderów pod względem każdej metryki. Dodatkowo można zauważyć, że alternatywny sposób oceny modeli, który został zaproponowany w pracy ocenia modele znacznie pozytywniej niż klasyczny sposób oceny. Wynika to z faktu, że anomalia jest uznawana za znalezioną w przypadku jeśli model wskaże jej moment występowania 10 minut wcześniej lub 10 minut później, zatem liczba prawdziwych pozytywów jest większa niż podczas klasycznego sposobu oceny. Z tego powodu precyzja, TPR oraz wynik F1 przyjmują większe wartości.

4.1.1.1 Sieci typu AutoEncoder

Na podstawie wyników zawartych w Tabeli 4.1, można wywnioskować, że wśród AutoEncoderów w ramach alternatywnego sposobu oceny najlepszą siecią okazała się sieć LSTM, która osiągnęła TPR równy 0.1935, precyzję równą 0.1403 oraz wynik F1 na poziomie 0.1627. Sieć LSTM osiągnęła lepsze wyniki niż nawet model głosujący, który co prawda miał trochę lepszą precyzję, lecz gorszy TPR oraz wynik F1. Najgorzej natomiast poradziła sobie sieć CNN, która nie dość, że wykrywała najmniej anomalii, to jeszcze zgłaszała najwięcej fałszywych pozytywów, o czym świadczy najniższa precyzja oraz TPR równe odpowiednio 0.0971 i 0.1613. Patrząc na-

Alternatywny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Uczenie chronologiczne		
	TPR	precision	F1	TPR	precision	F1
CNN	0.1613	0.0971	0.1212	0.1471	0.0170	0.0307
LSTM	0.1935	0.1403	0.1627	0.4706	0.0198	0.0380
CNN LSTM	0.1935	0.1311	0.1563	0.4117	0.0216	0.0410
CNN LSTM v2	0.1855	0.1329	0.1549	0.4412	0.0239	0.0454
Model głosujący	0.1855	0.1447	0.1625	0.1855	0.0241	0.0455

Tablica 4.1: Skuteczność modeli sieci typu AutoEncoder liczona alternatywnym sposobem oceny.

tomiast na wyniki w Tabeli 4.2 można zauważyć, że zgodnie z przypuszczeniami wartości TPR, precyzji oraz wyniku F1 są niższe, gdyż w klasycznym sposobie oceny model musi oznaczyć dokładny moment wystąpienia anomalii, by była ona uznana za prawdziwy pozytyw. W klasycznym sposobie ewaluacji najlepszy okazał się model głosujący z TPR 0.1452, precyzją 0.0690 oraz wynikiem F1 0.0935. Pomimo tego, że sieć CNN LSTM v2 miała większą precyzję równą 0.0698, to była to jednak nieznaczna różnica, w porównaniu z niższym TPR oraz wynikiem F1 sieci CNN LSTM v2 wynoszącym odpowiednio 0.1210 oraz 0.0885.

4.1.1.2 Sieci przewidujące

Na podstawie Tabeli 4.3 można odnotować, że wśród modeli predykcyjnych najlepszy wynik F1 wynoszący 0.3846 osiągnął model głosujący, a najwyższy współczynnik prawdziwych pozytywów równy 0.6389 miała sieć CNN LSTM. Najgorzej poradził sobie model CNN, który miał najniższy TPR równy 0.5714 oraz precyzję wynoszącą 0.2381. Zarówno model głosujący jak i sieć CNN LSTM poradziły sobie bardzo dobrze, sieć CNN LSTM zidentyfikowała więcej anomalii, lecz model głosujący rzadziej zgłaszał fałszywe pozytywy, więc w zależności od tego, która cecha

Klasyczny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Uczenie chronologiczne		
	TPR	precision	F1	TPR	precision	F1
CNN	0.1048	0.0570	0.0739	0.0882	0.0099	0.0179
LSTM	0.1129	0.0664	0.0836	0.4118	0.0157	0.0303
CNN LSTM	0.1129	0.0645	0.0821	0.3824	0.0181	0.0346
CNN LSTM v2	0.1210	0.0698	0.0885	0.3824	0.0187	0.0356
Model głosujący	0.1452	0.0690	0.0935	0.3824	0.0200	0.0380

Tablica 4.2: Skuteczność modeli sieci typu AutoEncoder liczona klasycznym sposobem oceny.

modelu jest bardziej pożądana, to wtedy można uznać jeden z modeli za lepszy. W przypadku zastosowania produkcyjnego pożądana jest jak największa precyzja, lecz różnica w precyzji między modelami wynosi tylko 0.0193 podczas, gdy różnica w TPR to 0.0674. Zatem wśród modeli ciężko wyłonić najlepszą sieć.

W Tabeli 4.4 można zauważyć, że również w przypadku sieci predykcyjnych osiągają one niższe wyniki w klasycznym sposobie oceny niż w alternatywnym sposobie oceny. Największemu, bo ponad dwukrotnemu spadkowi uległa precyzja (przykładowo dla modelu głosującego z 0.2899 do 0.1169). W klasycznym sposobie oceny najlepiej wypadł model głosujący uzyskując najwyższą precyzję oraz wynik F1 na poziomie 0.1169 oraz 0.1765 odpowiednio, a także TPR równy 0.3600.

Alternatywny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Uczenie chronologiczne		
	TPR	precision	F1	TPR	precision	F1
CNN	0.5714	0.2381	0.3361	0.5882	0.0356	0.0671
LSTM	0.6111	0.2750	0.3793	0.6471	0.0416	0.0782
CNN LSTM	0.6389	0.2706	0.3802	0.8236	0.0280	0.0542
Model głosujący	0.5715	0.2899	0.3846	0.5882	0.0411	0.0768

Tablica 4.3: Skuteczność modeli sieci przewidujących liczona alternatywnym sposobem oceny.

4.1.2. Gaussian dropout oraz mechanizm self attention

Wyniki uzyskane w sekcji 4.1.1 nie były jednak idealne i w ramach pracy magisterskiej dokonano starań, by wyniki modeli były jak najlepsze. Pierwszym krokiem dążącym do poprawy

Klasyczny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Uczenie chronologiczne		
	TPR	precision	F1	TPR	precision	F1
CNN	0.3600	0.0968	0.1525	0.3824	0.0212	0.0403
LSTM	0.3600	0.1023	0.1593	0.4706	0.0278	0.0524
CNN LSTM	0.4000	0.1064	0.1681	0.7059	0.0213	0.0413
Model głosujący	0.3600	0.1169	0.1765	0.3824	0.0246	0.0462

Tablica 4.4: Skuteczność modeli sieci przewidujących liczona klasycznym sposobem oceny.

modeli była lekka modyfikacja architektur sieci polegająca na zamienieniu warstwy Dropout na warstwę GaussianDropout oraz dodanie do sieci warstw implementujących mechanizm self attention [3]. Pozwoliło to na poprawę skuteczności modeli, z czego wpływ zmiany był bardziej widoczny w przypadku sieci przewidujących. Zmodyfikowane architektury sieci widoczne są na Rysunkach w dodatku w rozdziale 6.2, a ich wyniki zaprezentowano w Tabelach 4.5, 4.6, 4.7, 4.8.

4.1.2.1 Sieci typu AutoEncoder

Na podstawie Tabeli 4.5 można odnotować wzrost precyzji modeli po użyciu warstw Gaussian dropout oraz warstw self attention. Wynik F1 poprawił się dla sieci CNN, CNN LSTM v2 oraz modelu głosującego, lecz spadł dla sieci LSTM oraz CNN LSTM. Natomiast współczynnik prawdziwych pozytywów poprawił się dla sieci CNN oraz CNN LSTM. Pod względem precyzji nie poprawiła się jedynie sieć CNN LSTM. W tym przypadku również ciężko wskazać jest najlepszą sieć. Pod względem liczby wykrytych anomalii najlepszy jest model CNN LSTM v2, który uzyskał TPR równy 0.1935, co jest lepszym wynikiem w porównaniu do TPR równego 0.1694 dla modelu głosującego. Jednakże model głosujący wykazał się lepszą precyzją wynoszącą 0.1579 w porównaniu do precyzji 0.1404 uzyskanej przez model CNN LSTM v2.

W przypadku klasycznego sposobu oceny (widocznego w Tabeli 4.6) wszystkie modele oprócz sieci LSTM osiągnęły lepszy wynik F1 niż modele podstawowe. Jako najlepsze sieci można uznać sieć CNN LSTM v2 oraz model głosujący. Sieć CNN LSTM przoduje pod względem TPR z wynikiem 0.1210, natomiast model głosujący ma najlepszą precyzję równą 0.0844. Oba modele mają zbliżony wynik F1, który wynosi 0.0893 dla sieci CNN LSTM v2 oraz 0.0935 dla modelu głosującego.

Alternatywny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Uczenie chronologiczne		
	TPR	precision	F1	TPR	precision	F1
CNN	0.1694	0.1148	0.1368	0.1765	0.0195	0.0352
LSTM	0.1855	0.1322	0.1544	0.5883	0.0195	0.0378
CNN LSTM	0.1855	0.1211	0.1465	0.3824	0.0205	0.0389
CNN LSTM v2	0.1935	0.1404	0.1627	0.4412	0.0203	0.0388
Model głosujący	0.1694	0.1579	0.1634	0.2941	0.0287	0.0522

Tablica 4.5: Skuteczność ulepszonych modeli sieci typu AutoEncoder liczona alternatywnym sposobem oceny.

Klasyczny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Uczenie chronologiczne		
	TPR	precision	F1	TPR	precision	F1
CNN	0.1048	0.0631	0.0788	0.1176	0.0125	0.0227
LSTM	0.1210	0.0704	0.0890	0.4706	0.0142	0.0275
CNN LSTM	0.1129	0.0619	0.0800	0.3529	0.0173	0.0330
CNN LSTM v2	0.1210	0.0710	0.0893	0.3824	0.0158	0.0303
Model głosujący	0.1048	0.0844	0.0935	0.3529	0.0198	0.0374

Tablica 4.6: Skuteczność ulepszonych modeli sieci typu AutoEncoder liczona klasycznym sposobem oceny.

4.1.2.2 Sieci przewidujące

Biorąc pod uwagę alternatywny oraz klasyczny sposób oceny, prawie wszystkie sieci uzyskały lepszy wynik F1 oraz poprawiły swoją skuteczność (Tabele 4.7, 4.8). Jedynie sieć CNN LSTM w klasycznym sposobie oceny pogorszyła się pod względem tych parametrów.

W przypadku alternatywnego sposobu oceny jednoznacznie najlepiej sobie poradził model głosujący, który co prawda uzyskał TPR równy 0.6111, czyli o 1 punkt procentowy niższy od modelu CNN LSTM. Jednakże precyzja oraz wynik F1 modelu głosującego wynoszące odpowiednio 0.3143 i 0.4151 są wyższe niż wyniki pozostałych sieci.

W klasycznym trybie oceny, wszystkie modele osiągnęły taki sam TPR równy 0.3600, zatem wyłonienie najlepszego modelu zostało dokonane na podstawie porównania precyzji oraz wyników F1 modeli. Pod względem tych metryk najlepiej poradził sobie model głosujący, który uzyskał

precyzję na poziomie 0.1169 oraz wynik F1 równy 0.1765.

Alternatywny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Uczenie chronologiczne		
	TPR	precision	F1	TPR	precision	F1
CNN	0.6111	0.2750	0.3793	0.5294	0.0341	0.0641
LSTM	0.6111	0.2821	0.3860	0.4412	0.0330	0.0613
CNN LSTM	0.6216	0.2771	0.3833	0.7941	0.0274	0.0529
Model głosujący	0.6111	0.3143	0.4151	0.5588	0.0336	0.0635

Tablica 4.7: Skuteczność ulepszonych modeli sieci przewidujących liczona alternatywnym sposobem oceny.

Klasyczny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Uczenie chronologiczne		
	TPR	precision	F1	TPR	precision	F1
CNN	0.3600	0.1034	0.1607	0.4706	0.0274	0.0518
LSTM	0.3600	0.1047	0.1622	0.2648	0.0183	0.0343
CNN LSTM	0.3600	0.0978	0.1538	0.5882	0.0185	0.0358
Model głosujący	0.3600	0.1169	0.1765	0.4706	0.0258	0.0489

Tablica 4.8: Skuteczność ulepszonych modeli sieci przewidujących liczona klasycznym sposobem oceny.

4.1.3. Próba usunięcia trendu z danych

Kolejna modyfikacja podejścia, która potencjalnie mogła poprawić działanie modeli, dotyczyła zmiany przetwarzania danych wejściowych. Zauważono, że wartości cech dla pomiarów zmieniają się z czasem przez co wartości cech występujące w ostatnich 20% okien czasowych nie pojawiają się w pierwszych 80% okien czasowych. Może to być potencjalną przyczyną tego, że sieci słabo sobie radzą w przypadku gdy okna czasowe są dzielone na zbiór treningowy oraz testowy chronologicznie. W celu ominięcia tej przeszkody wykorzystano podejście, które pozwalało nam pozbyć się zmieniającego trendu z danych. Polegało to na tym, że oryginalne zadanie zostało zamienione:

- w przypadku sieci przewidujących – na zadanie predykcji tego jak wartość pomiaru się zmieni w danym momencie na podstawie informacji jak wartość pomiaru zmieniała się w kilku poprzednich krokach,

- w przypadku sieci typu AutoEncoder – na skompresowaniu a potem odtworzeniu kilku kolejnych wartości reprezentujących zmianę wartości pomiaru dla kilku kolejnych kroków czasowych.

Zmiana wartości pomiaru dla *k*-tego kroku czasowego obliczana jest jako różnica wartości pomiaru w *k*-tym oraz *k-1* kroku czasowym. Podejście to jednak nie odniosło spodziewanych rezultatów. Wręcz przeciwnie skuteczność modeli zarówno przewidujących jak i modeli typu AutoEncoder spadła, zatem to podejście zostało porzucone.

4.1.4. Online learning

Kolejnym etapem poprawy skuteczności modeli pod względem ich działania w przypadku klasycznego dzielenia danych było zastosowanie online learningu. Polega to na tym, że sieci oprócz przewidywania/odtwarzania nowych danych douczają się na nich. Podczas testów okazało się, że sieci odnotowują największą poprawę w przypadku, gdy sieci otrzymują 100 okien czasowych do predykcji/odtworzenia, a następnie douczają się na tej partii danych przez 10 epok i właśnie taki tryb uczenia online został zastosowany. Przetestowana została również opcja uczenia online w której po predykcji/odtworzeniu porcji nowych danych zawierających k okien czasowych ze zbioru treningowego były usuwane najstarsze k okien czasowych, a następnie do zbioru treningowego była dopisywana nowa porcja danych. Następnie sieć douczała się przez kilka epok na uaktualnionym zbiorze treningowym. W tym przypadku również odnotowano poprawę skuteczności modeli, jednak opisany wcześniej sposób z douczaniem sieci jedynie na nowych danych osiągnął lepsze rezultaty, które zostały pokazane w Tabelach poniżej.

4.1.4.1 Sieci typu AutoEncoder

Na podstawie poniższych Tabeli 4.9, 4.10 można zaobserwować, że zastosowanie online learningu zaowocowało poprawą precyzji oraz wyniku F1 wszystkich modeli w porównaniu do uczenia chronologicznego w ramach alternatywnego sposobu oceny. Jeśli chodzi o klasyczny sposób oceny, to większość sieci również poprawiła swój wynik F1 oraz precyzję.

W alternatywnym trybie oceny najlepiej poradził sobie model głosujący uzyskując TPR równy 0.3529, precyzję równą 0.0895 oraz wynik F1 0.0584. Rezultat ten cechuje się czterokrotnie wyższą precyzją niż w przypadku uczenia chronologicznego bez użycia metody online learningu.

Jeśli weźmie się pod uwagę klasyczny tryb ewaluacji, to w tym przypadku za najlepszy model można również uznać model głosujący. Osiągnął on najlepsze wyniki pod względem każdej metryki wśród wszystkich modeli uzyskując TPR 0.2353, precyzję 0.0211, wynik F1 0.0387.

Alternatywny sposób oceny			
Rodzaj modelu	Online learning		
	TPR	precision	F1
CNN	0.1471	0.0227	0.0394
LSTM	0.3529	0.0302	0.0556
CNN LSTM	0.1765	0.0213	0.0380
CNN LSTM v2	0.2059	0.0223	0.0402
Model głosujący	0.3529	0.0895	0.0584

Tablica 4.9: Skuteczność ulepszonych modeli sieci typu AutoEncoder podczas uczenia w trybie online learningu liczona alternatywnym sposobem oceny.

Klasyczny sposób oceny			
Rodzaj modelu	Online learning		
	TPR	precision	F1
CNN	0.0882	0.0131	0.0228
LSTM	0.2059	0.0162	0.0300
CNN LSTM	0.1176	0.0134	0.0241
CNN LSTM v2	0.1176	0.0119	0.0216
Model głosujący	0.2353	0.0211	0.0387

Tablica 4.10: Skuteczność ulepszonych modeli sieci typu AutoEncoder podczas uczenia w trybie online learningu liczona klasycznym sposobem oceny.

4.1.4.2 Sieci przewidujące

Po zastosowaniu online learningu większość sieci predykcyjnych osiągnęło lepszy wynik F1 zarówno w alternatywnym jak i klasycznym sposobie oceny, co widać w Tabelach 4.11, 4.12.

W przypadku alternatywnego trybu oceny najlepszą siecią okazała się sieć CNN, która uzyskała najlepszy TPR wynoszący 0.4118 oraz precyzję równą 0.4003 i wynik F1 na poziomie 0.0735.

Sieć CNN wypadła najlepiej również, w przypadku oceny modeli klasycznym sposobem. Pod każdym względem osiągnęła ona najlepsze wyniki spośród modeli, a były to TPR 0.2941, precyzja 0.0260 oraz wynik F1 0.0477.

Alternatywny sposób oceny			
Rodzaj modelu	Online learning		
	TPR	precision	F1
CNN	0.4118	0.0403	0.0735
LSTM	0.3529	0.0385	0.0694
CNN LSTM	0.4118	0.0286	0.0534
Model głosujący	0.3529	0.0414	0.0741

Tablica 4.11: Skuteczność ulepszonych modeli sieci przewidujących podczas uczenia w trybie online learningu liczona alternatywnym sposobem oceny.

Klasyczny sposób oceny			
Rodzaj modelu	Online learning		
	TPR	precision	F1
CNN	0.2941	0.0260	0.0477
LSTM	0.2059	0.0211	0.0384
CNN LSTM	0.2353	0.0151	0.0284
Model głosujący	0.2059	0.0227	0.0408

Tablica 4.12: Skuteczność ulepszonych modeli sieci przewidujących podczas uczenia w trybie online learningu liczona klasycznym sposobem oceny.

4.1.5. Statystyka zmian

Żeby modele mogły zostać użyte w środowisku produkcyjnym nie powinny za często alarmować o nieistniejącym problemie, zatem kolejne próby poprawy skuteczności modeli skupiły się na zwiększeniu precyzji modelu, żeby nie zgłaszał on zbyt wielu fałszywych pozytywów. Warto zauważyć, że w 24 godzinnym cyklu można zaobserwować momenty, w których zmiany w pomiarach są duże. Mogą one wynikać z momentów większego obciążenia przykładowo, gdy pracownicy rozpoczynają pracę w fabryce papieru. Analogicznie serwery sieciowe również notują zwiększone obciążenie w momencie gdy użytkownicy z rana zaczynają korzystać z ich usług. W tamtych chwilach nagłe zmiany w pomiarach nie są anomaliami, tylko standardowymi zmianami wynikającymi z cyklu dnia. W celu uniknięcia zgłaszania przez model anomalii w tych chwilach wykorzystana została statystyka zmian (opisana w rozdziale 3.1.6). Została ona utworzona na podstawie błędów predykcji/odtworzenia na zbiorze treningowym. Statystyka ta następnie może zostać podzielona na k minutowe koszyki. Jeśli model wykryje anomalie to dodatkowo spraw-

dza, do którego z koszyków w statystyce zmian należy przewidywane/odtwarzane okno czasowe. Jeśli wartość błędu, który zwraca model dla okna czasowego nie przekracza pewnego percentylu wartości błędów dla koszyka w statystyce zmian pomnożonego przez pewien współczynnik, to można uznać, że model nie wykrył anomalii, gdyż w tym przedziale czasowym obserwowane są większe zmiany. Pozwala to na znaczne zmniejszenie liczby fałszywych pozytywów, co widać na podstawie wzrostu precyzji modeli w Tabelach poniżej. Podczas testów okazało się, że wartość błędu dla okna czasowego najlepiej porównywać z 98 percentylem wartości błędów z koszyka w statystyce zmian pomnożonego przez 1. Natomiast koszyk w statystyce zmian zajmował 10 minutowy przedział czasu.

4.1.5.1 Sieci typu AutoEncoder

Na podstawie Tabeli 4.13, 4.14, można wywnioskować, że w przypadku alternatywnego oraz klasycznego sposobu oceny zastosowanie statystyki zmian znacząco poprawiło precyzję modeli. Co prawda wskaźnik prawdziwych pozytywów uległ pogorszeniu, lecz większość modeli osiągnęło lepszy wynik F1.

Za najlepszą sieć można uznać model głosujący, który uzyskał TPR równy 0.1532, precyzję równą 0.4130 oraz wynik F1 na poziomie 0.2235. Pozostałe modele miały niższy TPR oraz zazwyczaj mniejszą precyzję.

Model głosujący wypadł również najlepiej w klasycznym trybie oceny. Uzyskał on TPR równy 0.1049, precyzję 0.1733 oraz wynik F1 0.1307. Pozostałe modele miały niższe wyniki w każdej wymienionej metryce.

Alternatywny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Online learning		
	TPR	precision	F1	TPR	precision	F1
CNN	0.1210	0.3659	0.1818	0.1176	0.0186	0.0321
LSTM	0.1290	0.3077	0.1818	0.2941	0.0265	0.0485
CNN LSTM	0.1210	0.4286	0.1887	0.1765	0.0205	0.0367
CNN LSTM v2	0.1452	0.3750	0.2093	0.2059	0.0208	0.0378
Model głosujący	0.1532	0.4130	0.2235	0.3236	0.0314	0.0573

Tablica 4.13: Skuteczność ulepszonych modeli sieci typu AutoEncoder z użyciem statystyki zmian liczona alternatywnym sposobem oceny.

Klasyczny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Online learning		
	TPR	precision	F1	TPR	precision	F1
CNN	0.0484	0.1304	0.0706	0.0882	0.0135	0.0233
LSTM	0.0565	0.1228	0.0773	0.1471	0.0122	0.0226
CNN LSTM	0.0484	0.1579	0.0741	0.1176	0.0127	0.0230
CNN LSTM v2	0.0726	0.1636	0.1006	0.1176	0.0110	0.0202
Model głosujący	0.1049	0.1733	0.1307	0.2647	0.0211	0.0392

Tablica 4.14: Skuteczność ulepszonych modeli sieci typu AutoEncoder z użyciem statystyki zmian liczona klasycznym sposobem oceny.

4.1.5.2 Sieci przewidujące

Tabele 4.15, 4.16 pokazują, że wszystkie modele poprawiły swoją precyzję oraz wynik F1 w przypadku obu metod oceny. Nieznacznemu pogorszeniu uległ jednak wskaźnik prawdziwych pozytywów sieci.

Na podstawie Tabeli 4.15 można wyznaczyć dwa najlepsze modele. Sieć CNN LSTM, która uzyskała TPR 0.5882, precyzję 0.3175, wynik F1 0.4124 oraz model głosujący z wynikami takimi jak TPR 0.5313, precyzja 0.3696 i wynik F1 0.4359. Co prawda sieć CNN LSTM znalazła więcej anomalii (o czym świadczy wyższy TPR), to jednak model głosujący rzadziej zgłaszał fałszywe pozytywy (gdyż uzyskał większą precyzję), a celem zastosowania statystyki zmian była redukcja liczby fałszywych alarmów, zatem za lepszy model można uznać model głosujący.

W klasycznym trybie ewaluacji najlepiej wypadł model głosujący. Miał on TPR równy 0.2800, precyzję równą 0.1458 oraz wynik F1 wynoszący 0.1981. Były to najlepsze wyniki pod względem precyzji oraz wyniku F1. Jedynie pod względem TPR jeden z modeli, sieć CNN LSTM, poradziła sobie lepiej uzyskując wynik 0.3200.

4.1.6. Piecewise Aggregate Approximation

Ostatnią próbą poprawienia skuteczności modeli było zastosowanie na zbiorze danych metody Piecewise Aggregate Approximation (rozdział 3.1.7). W trakcie badań najlepsze wyniki zostały osiągnięte poprzez zastąpienie każdych 2 przyległych pomiarów jednym pomiarem będącym ich średnią. Jednak wyniki tego eksperymentu były niejednoznaczne. Niektóre z architektur odnotowały drobną poprawę skuteczności, podczas gdy inne odnotowały lekki spadek skuteczności.

Alternatywny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Online learning		
	TPR	precision	F1	TPR	precision	F1
CNN	0.5313	0.3091	0.3908	0.3529	0.0490	0.0860
LSTM	0.5588	0.2923	0.3838	0.3529	0.0455	0.0805
CNN LSTM	0.5882	0.3175	0.4124	0.3529	0.0337	0.0615
Model głosujący	0.5313	0.3696	0.4359	0.3235	0.0480	0.0837

Tablica 4.15: Skuteczność ulepszonych modeli sieci przewidujących z użyciem statystyki zmian liczona alternatywnym sposobem oceny.

Klasyczny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Online learning		
	TPR	precision	F1	TPR	precision	F1
CNN	0.2800	0.1228	0.1707	0.1765	0.0229	0.0405
LSTM	0.2800	0.1045	0.1522	0.1765	0.0211	0.0376
CNN LSTM	0.3200	0.1212	0.1758	0.1471	0.0132	0.0242
Model głosujący	0.2800	0.1458	0.1918	0.1471	0.0205	0.0360

Tablica 4.16: Skuteczność ulepszonych modeli sieci przewidujących z użyciem statystyki zmian liczona klasycznym sposobem oceny.

4.1.6.1 Sieci typu AutoEncoder

Tabele 4.17, 4.18 pokazują, że poprawie uległa precyzja pojedynczych sieci. Jedynie model głosujący odnotował spadek precyzji oraz wzrost współczynnika prawdziwych pozytywów. Jednakże wynik F1 pogorszył się dla wszystkich sieci.

Za najlepszy model na podstawie Tabeli 4.17 można uznać model LSTM, który miał TPR równy 0.1129, najwyższą precyzję równą 0.5186 oraz wynik F1 równy 0.1854. Model głosujący miał co prawda lepszy TPR równy 0.1694, to jednak uzyskał on niską precyzję w porównaniu do LSTM równą 0.2500.

W przypadku klasycznego sposobu oceny wybór najlepszego modelu nie jest trywialny. Jednak największą wagę należy przyznać metryce precyzji, gdyż celem zastosowania metody Piecewise Aggregate Approxmiation było właśnie poprawienie modeli, by zgłaszały mniej fałszywych pozytywów. W takim wypadku najlepsze okazały się modele LSTM oraz CNN LSTM, które osiągnęły dokładnie takie same wyniki TPR 0.0565, precyzję 0.2258 oraz wynik F1 0.0903.

Alternatywny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Online learning		
	TPR	precision	F1	TPR	precision	F1
CNN	0.0968	0.3750	0.1538	0.1176	0.0214	0.0362
LSTM	0.1129	0.5186	0.1854	0.2941	0.0252	0.0465
CNN LSTM	0.1048	0.4483	0.1699	0.2059	0.0243	0.0435
CNN LSTM v2	0.1129	0.4242	0.1783	0.2941	0.0292	0.0532
Model głosujący	0.1694	0.2500	0.2019	0.3236	0.0311	0.0567

Tablica 4.17: Skuteczność ulepszonych modeli sieci typu AutoEncoder z użyciem techniki Piecewise Aggregate Approximation liczona alternatywnym sposobem oceny.

Klasyczny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Online learning		
	TPR	precision	F1	TPR	precision	F1
CNN	0.4032	0.1471	0.0633	0.0882	0.0154	0.0262
LSTM	0.0565	0.2258	0.0903	0.1765	0.0143	0.0264
CNN LSTM	0.0565	0.2258	0.0903	0.1471	0.0167	0.0300
CNN LSTM v2	0.0645	0.2000	0.0976	0.2059	0.0190	0.0348
Model głosujący	0.0806	0.1311	0.1300	0.2941	0.0209	0.0390

Tablica 4.18: Skuteczność ulepszonych modeli sieci typu AutoEncoder z użyciem techniki Piecewise Aggregate Approximation liczona klasycznym sposobem oceny.

4.1.6.2 Sieci przewidujące

Na podstawie Tabeli 4.20, 4.19 można zaobserwować, że sieci predykcyjne również odnotowały wzrost precyzji. Jednakże w przeciwieństwie do AutoEncoderów, niektóre sieci predykcyjne uzyskały również lepszy TPR oraz wynik F1. Większą poprawę można zaobserwować w przypadku klasycznego sposobu oceny.

W Tabeli 4.20 najlepszym modelem był model LSTM, który wykrywał ponad połowę anomalii uzyskując TPR równy 0.5429, a co trzecia zgłaszana anomalia była rzeczywista o czym świadczy precyzja równa 0.3220.

Natomiast w klasycznym trybie ewaluacji najlepsza okazał się sieć CNN. Osiągnęła ona TPR równy 0.3846, precyzję równą 0.1754 oraz wynik F1 równy 0.2410. Pozostałe modele wypadły pod każdym względem gorzej.

Alternatywny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Online learning		
	TPR	precision	F1	TPR	precision	F1
CNN	0.5313	0.3148	0.3953	0.2941	0.0397	0.0699
LSTM	0.5429	0.3220	0.4043	0.3824	0.0410	0.0741
CNN LSTM	0.5588	0.2969	0.3878	0.4118	0.0390	0.0712
Model głosujący	0.4688	0.3488	0.4000	0.4412	0.0393	0.0721

Tablica 4.19: Skuteczność ulepszonych modeli sieci przewidujących z użyciem techniki Piecewise Aggregate Approximation liczona alternatywnym sposobem oceny.

Klasyczny sposób oceny						
Rodzaj modelu	Alternatywny podział danych			Online learning		
	TPR	precision	F1	TPR	precision	F1
CNN	0.3846	0.1754	0.2410	0.1765	0.0222	0.0395
LSTM	0.3333	0.1429	0.2000	0.2059	0.0199	0.0364
CNN LSTM	0.3704	0.1449	0.2083	0.2647	0.0224	0.0413
Model głosujący	0.3077	0.1739	0.2222	0.1538	0.0238	0.0412

Tablica 4.20: Skuteczność ulepszonych modeli sieci przewidujących z użyciem techniki Piecewise Aggregate Approximation liczona klasycznym sposobem oceny.

4.1.7. Podsumowanie

Na podstawie przeprowadzonych testów można porównać uzyskanie wyniki z rezultatami przedstawionymi w pracy [15]. Żeby porównanie było jak najbardziej wiarygodnie wybrane zostały z każdego podejścia najlepsze modele oceniane klasycznym sposobem oceny, które uczyły się na danych podzielonych alternatywnym sposobem z racji, że takie samo podejście do danych zastosowali autorzy pracy [15]. Porównanie modeli zostało przedstawione w Tabeli 4.21, a za najlepszy można uznać model predykcyjny CNN z Tabeli 4.20, który osiągnął TPR równy 0.3864, precyzję wynoszącą 0.1754 oraz wynik F1 równy 0.2410. Porównując model z wynikami uzyskanymi przez badaczy, tj. FPR (ang. False Positive Rate) równy 0.026, precyzja wynosząca 0.071 oraz wynik F1 równy 0.114 można stwierdzić, że uzyskany w niniejszej pracy model jest dwukrotnie lepszy. Wyższa precyzja przekłada się na fakt, że model uzyskany w pracy magisterskiej rzadziej zgłasza fałszywe alarmy, a wyższy wynik F1 świadczy o tym, że model w ogólności radzi sobie lepiej z identyfikowaniem anomalii niż podejście badaczy z pracy [15]. Można zatem

Rodzaj modelu	Typ sieci	Użyte podejście	TPR	precision	F1
Model z pracy [15]	–	–	–	0.071	0.114
Model głosujący	AutoEncoder	podejście podstawowe	0.1452	0.0690	0.0935
Model głosujący	predykcyjny	podejście podstawowe	0.3600	0.1169	0.1765
Sieć CNN LSTM v2	AutoEncoder	dodanie warstw Gaussian dropout i self attention	0.1210	0.0710	0.0893
Model głosujący	AutoEncoder	dodanie warstw Gaussian dropout i self attention	0.1048	0.0844	0.0935
Model głosujący	predykcyjny	dodanie warstw Gaussian dropout i self attention	0.3600	0.1169	0.1765
Model głosujący	AutoEncoder	zastosowanie statystyki zmian	0.1049	0.1733	0.1307
Model głosujący	predykcyjny	zastosowanie statystyki zmian	0.2800	0.1458	0.1918
Sieć LSTM	AutoEncoder	zastosowanie metody Piecewise Aggregate Approximation	0.0565	**0.2258**	0.0903
Sieć CNN LSTM	AutoEncoder	zastosowanie metody Piecewise Aggregate Approximation	0.0565	**0.2258**	0.0903
Sieć CNN	predykcyjny	zastosowanie metody Piecewise Aggregate Approximation	**0.3846**	0.1754	**0.2410**

Tablica 4.21: Porównanie wyników najlepszych modeli wykorzystanych w różnych etapach pracy magisterskiej z wynikami z pracy [15]. Modele zostały ocenione klasycznym sposobem.

stwierdzić, że metoda zastosowana w pracy magisterskiej jest jak najbardziej skuteczna, gdyż zdecydowanie lepiej się sprawdza w porównaniu z metodą znalezioną w literaturze.

Warto również zauważyć, że nie tylko model używający finalnego podejścia osiąga lepsze wyniki niż te uzyskane w pracy [15]. W przypadku sieci predykcyjnych nawet podstawowe podejście skutkuje lepszą precyzją oraz wynikiem F1. Natomiast sieci typu AutoEncoder prześcigają wyniki z pracy [15] dopiero po zastosowaniu statystyki zmian. Kolejnym wnioskiem wysuwającym się z analizy Tabeli 4.21 jest fakt, że w przypadku klasycznego trybu ewaluacji zazwyczaj najlepszą siecią był model głosujący, zmieniło się to dopiero w finalnym podejściu, gdzie to inne modele sieci takie jak CNN, LSTM, czy CNN LSTM okazały się najbardziej skuteczne. Dodatkowo można zauważyć, że używając tego samego podejścia, lepsze rezultaty uzyska się korzystając z sieci predykcyjnych. Osiągają one wyższy wskaźnik prawdziwych pozytywów oraz wynik F1 niż sieci typu AutoEncoder. Sieci AutoEncoder mają za to wyższą precyzję, lecz różnica w tej metryce jest mniejsza niż w przypadku TPR oraz wyniku F1.

Porównując modele pod względem alternatywnego sposobu oceny w Tabeli 4.22 za najlepszy model uznać można model głosujący predykcyjny w podejściu, w którym zastosowano statystykę zmian. Uzyskał on TPR równy 0.5313, precyzję równą 0.3696 oraz wynik F1 równy 0.4359. Można również zauważyć, że także w alternatywnym sposobie oceny, sieci predykcyjne radzą sobie lepiej od sieci typu AutoEncoder. Te pierwsze mają wyższy TPR oraz wynik F1, podczas gdy sieci typu AutoEncoder mają trochę większą precyzję.

Rodzaj modelu	Typ sieci	Użyte podejście	TPR	precision	F1
Sieć LSTM	AutoEncoder	podejście podstawowe	0.1935	0.1403	0.1627
Sieć CNN LSTM	predykcyjny	podejście podstawowe	**0.6389**	0.2706	0.3802
Model głosujący	predykcyjny	podejście podstawowe	0.5716	0.2899	0.3846
Sieć CNN LSTM v2	AutoEncoder	dodanie warstw Gaussian dropout i self attention	0.1935	0.1404	0.1627
Model głosujący	AutoEncoder	dodanie warstw Gaussian dropout i self attention	0.1694	0.1579	0.1634
Model głosujący	predykcyjny	dodanie warstw Gaussian dropout i self attention	0.6111	0.3143	0.4151
Model głosujący	AutoEncoder	zastosowanie statystyki zmian	0.1532	0.4130	0.2235
Model głosujący	predykcyjny	zastosowanie statystyki zmian	0.5313	0.3696	**0.4359**
Sieć LSTM	AutoEncoder	zastosowanie metody Piecewise Aggregate Approximation	0.1129	**0.5186**	0.1854
Sieć LSTM	predykcyjny	zastosowanie metody Piecewise Aggregate Approximation	0.5429	0.3220	0.4043
Model głosujący	predykcyjny	zastosowanie metody Piecewise Aggregate Approximation	0.4688	0.3488	0.4000

Tablica 4.22: Porównanie wyników najlepszych modeli wykorzystanych w różnych etapach pracy magisterskiej. Modele zostały ocenione alternatywnym sposobem.

4.2. Praca na danych z logserwera

Po tym jak oceniono działanie modeli na etykietowanym zbiorze danych i uzyskano ostateczne podejście do problemu, to można przejść teraz do etapu, w którym modele zostaną użyte w celu wykrywania anomalii na danych pobranych z logserwera firmy EMCA. Polegało to na weryfikacji anomalii zgłoszonych przez modele głosujący AutoEncoderów oraz model głosujący predykcyjny z sekcji 4.1.6. Na tej podstawie oceniona została precyzja każdego z modeli na zbiorze danych pochodzącym z logserwera. Trzeba jednak zwrócić uwagę na to, że na tym zbiorze danych nie można policzyć True Positive Rate oraz wyniku F1 modeli, gdyż zbiór nie jest etykietowany, a ręczne znalezienie wszystkich anomalii przez człowieka jest zbyt pracochłonne. Precyzja modeli została przedstawiona w Tabeli 4.23.

Precyzja modeli na zbiorze danych pochodzących z logserwera				
Modelu głosujący	Na podstawowym zbiorze		Na rozszerzonym zbiorze	
	liczba zgłoszonych anomalii	precision	liczba zgłoszonych anomalii	precision
AutoEncoder	46	0.5435	77	0.5064
Predykcyjny	25	0.5600	37	0.8982

Tablica 4.23: Precyzja modeli głosujących wykorzystujących podejście z sekcji 4.1.6 na zbiorze danych pochodzących z logserwera.

Okazało się, że modele uzyskały lepszą precyzję na danych pochodzących z logserwera firmy EMCA w porównaniu do precyzji na danych testowych. Za najlepszy model można uznać model

głosujący predykcyjny działający na rozszerzonym zbiorze danych ze względu na fakt, że osiągnął on precyzję równą 0.8982, czyli jedynie 1 na 10 zgłaszanych anomalii przez ten model jest fałszywym pozytywem, dzięki czemu taki model mógłby zostać wykorzystany w środowisku produkcyjnym, gdyż ekspert systemu nie traciłby wiele czasu na weryfikacje fałszywych alarmów. Dodatkowo można wyciągnąć następujące wnioski:

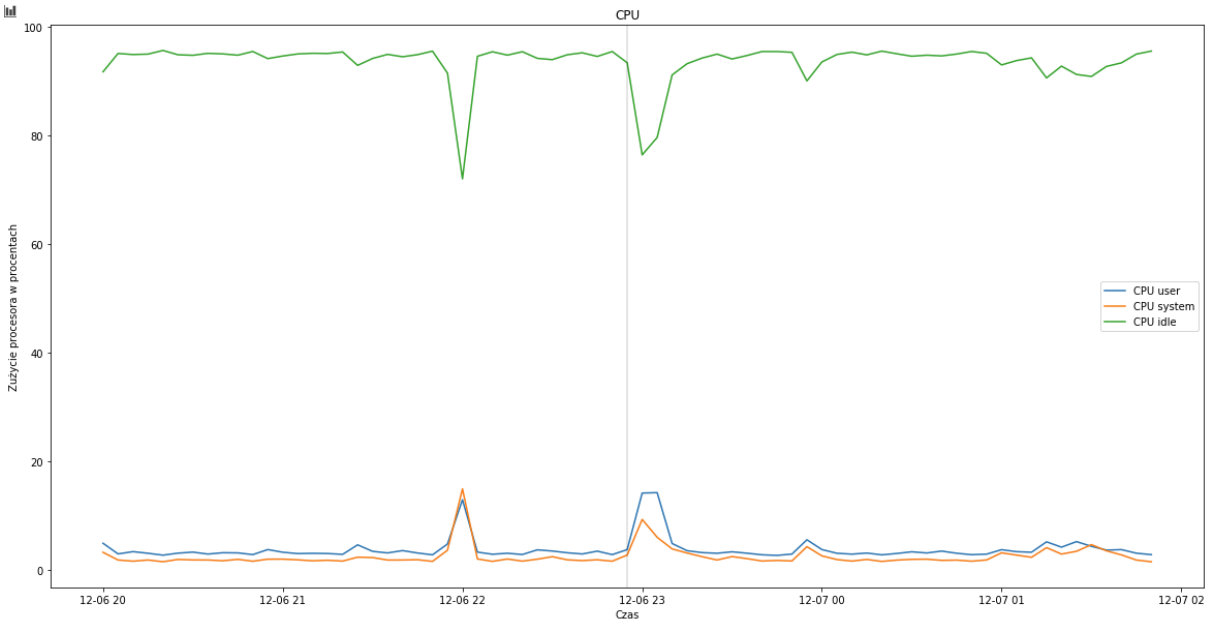
- odnośnie liczby zgłaszanych anomalii:
  - modele pracujące na rozszerzonym zbiorze zgłaszały więcej anomalii niż podczas pracy na podstawowym zbiorze,
  - model głosujący AutoEncoderów zgłaszał więcej anomalii niż model głosujący predykcyjny.
- odnośnie precyzji modeli głosujących:
  - model głosujący predykcyjny uzyskał lepszą precyzję niż model głosujący AutoEncoderów zarówno na podstawowym jak i rozszerzonym zbiorzem
  - w przeciwieństwie do modelu głosującego AutoEncoderów, który uzyskał lekko lepszą precyzję na zbiorze podstawowym niż rozszerzonym, model głosujący predykcyjny uzyskał znacznie lepszą precyzję (wynoszącą 0.8982) na zbiorze rozszerzonym niż na zbiorze podstawowym.

4.2.1. Wykrywane anomalie

Jeśli analizie podda się anomalie zgłaszane w zbiorze firmy EMCA przez modele, to okaże się, że zidentyfikowane są przede wszystkim 2 rodzaje anomalii. Pierwszy rodzaj anomalii wyróżnia się nagłym, krótkotrwałym wzrostem zużycia procesora oraz pamięci RAM (Rysunki 4.1, 4.2). Przykładowo na Rysunku 4.1 można zauważyć, że między godziną 20:00 dnia 6 grudnia 2020 roku, a godziną 02:00 dnia 7 grudnia 2020 roku procesor używał kilka procent mocy na procesy systemowe oraz procesy użytkownika. Jednakże w okolicach godziny 22:00 oraz godziny 23:00 wystąpiły 2 nagłe skoki w zużyciu procesora, gdy procesy systemowe oraz procesy użytkownika wykorzystywały razem ponad 20% mocy. Trwały one około 10 minut i jak widać na rysunku poprzez oznaczenie pionową szarą kreską w tym przypadku sieć poprawnie zidentyfikowała anomalie występującą około godziny 23:00. Natomiast na Rysunku 4.2 przedstawiony jest nagły skok zużycia pamięci RAM o tej samej porze, który zapewne był związany z zwiększeniem wykorzystania procesora widocznego na Rysunku 4.1.

Drugi natomiast typ anomalii zgłoszonych przez sieci polegał na wystąpieniu rekordów typu Warning lub Error w logach. Sytuacja ta jest zobrazowana na Rysunku 4.3. W dniu 16 marca

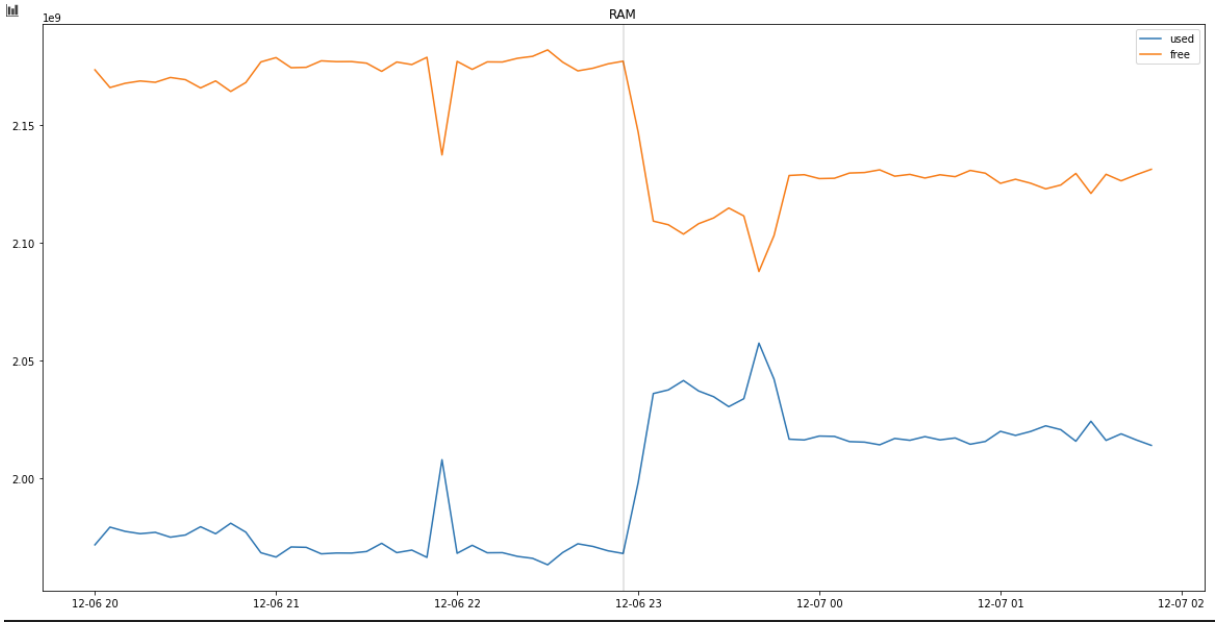
2021 roku można zaobserwować 2 nagłe wzrosty w występowaniu rekordów typu Warning w logach. Pierwszy wzrost rozpoczął się przed godziną 07:00 i trwał do godziny 08:00, natomiast drugi wzrost zaczął się około godziny 8:40 i zakończył się po godzinie 09:00. Pionową szarą kreską zaznaczony jest moment, w którym sieć zgłosiła ten niespodziewany wzrost rekordów typu Warning jako anomalie.



Rysunek 4.1: Przykład anomalii polegającej na nagłym wzroście zużycia CPU. Szara pionowa linia oznacza moment, który model zgłosił jako wystąpienie anomalii. (Opracowanie własne)

4.2.2. Porównanie finalnego podejścia z podejściem podstawowym

Warto również dokonać porównania jak ostateczne podejście z sekcji 4.1.6 sprawdza się w porównaniu do zastosowania podstawowych modeli z sekcji 4.1.1. W tym celu podstawowym modelom zlecono zadanie znalezienia anomalii w zbiorze EMCA, a następnie zgłoszone anomalie zostały zweryfikowane przez eksperta. Wyniki eksperymentu zostały przedstawione w Tabeli 4.24. Na pierwszy rzut oka można zauważyć, że modele te zgłaszały więcej anomalii niż modele wykorzystujące finalne podejście. Ponownie model głosujący AutoEncoderów zgłaszał więcej anomalii niż model głosujący predykcyjny, lecz tym razem oba modele głosujące uzyskały zbliżoną precyzję. Modele głosujące AutoEncoderów z podstawowego podejścia osiągnęły lekko niższą precyzję od modeli wykorzystujących podejście końcowe oraz zgłaszały one więcej anomalii, co powodowałoby sytuacje, w której ekspert musiałby weryfikować wiele fałszywych anomalii. W przypadku modelów predykcyjnych w szczególności na rozszerzonym zbiorze widoczna jest znacznie lepsza gorsza precyzja w podejściu podstawowym równa 0.5263 w porównaniu z precyzją 0.8982

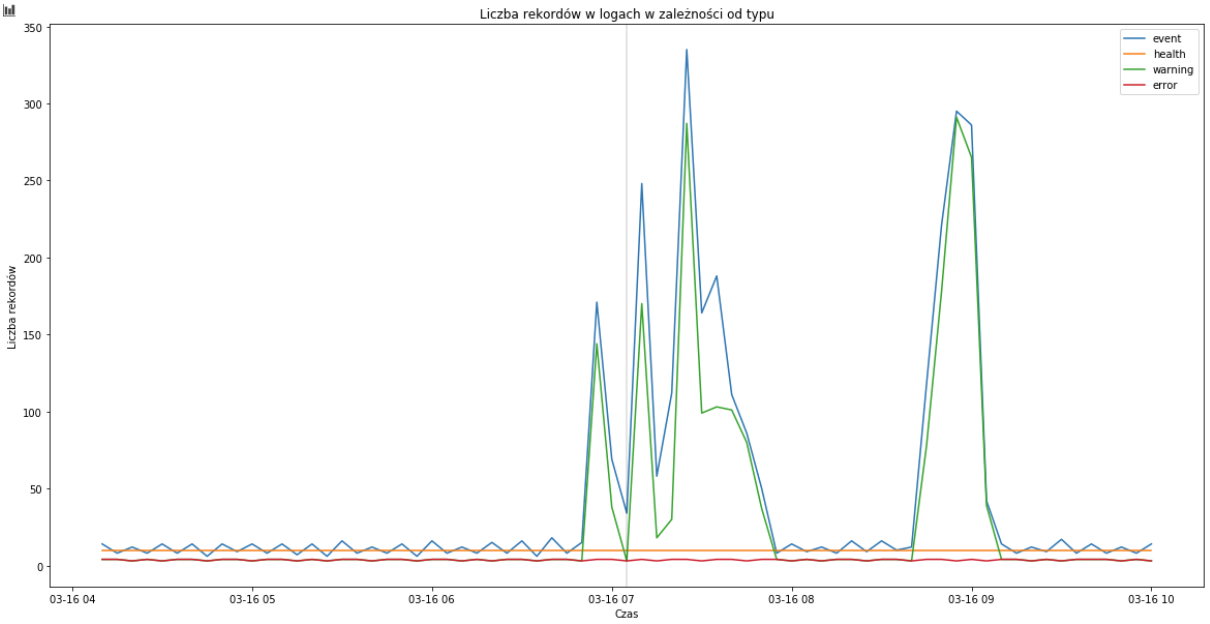


Rysunek 4.2: Przykład anomalii polegającej na nagłym wzroście zużycia pamięci RAM. Szara pionowa linia oznacza moment, który model zgłosił jako wystąpienie anomalii. (Opracowanie własne)

uzyskaną za pomocą wykorzystania finalnego podejścia. Można zatem z pewnością stwierdzić, że model predykcyjny w podejściu podstawowym jest gorszy, gdyż w jego przypadku co druga zgłaszana anomalia jest fałszywa podczas, gdy jedynie co dziesiąta anomalia jest fałszywa w przypadku jego odpowiednika z Tabeli 4.23. Nasuwa się zatem wniosek, że ostateczne podejście uzyskane w pracy, które wykorzystuje takie techniki jak:

- użycie warstw self attention,
- zastosowanie statystyki zmian,
- zastosowanie metody Piecewise Aggregate Approximation.

sprawdza się lepiej w zadaniu wykrywania anomalii niż podstawowe podejście, które nie wykorzysuje powyższych technik.



Rysunek 4.3: Przykład anomalii polegającej na wystąpieniu w logach rekordach typu Warning. Szara pionowa linia oznacza moment, który model zgłosił jako wystąpienie anomalii. (Opracowanie własne)

Precyzja modeli na zbiorze danych pochodzących z logserwera				
Rodzaj modelu	Na podstawowym zbiorze		Na rozszerzonym zbiorze	
	liczba zgłoszonych anomalii	precision	liczba zgłoszonych anomalii	precision
AutoEncodery	131	0.5038	146	0.5000
Predykcyjny	73	0.5479	95	0.5263

Tablica 4.24: Precyzja podstawowych modeli głosujących z sekcji 4.1.1 na zbiorze danych pochodzących z logserwera.

5. Podsumowanie

W pracy rozwiązany został problem wykrywania anomalii na podstawie logów zawierających wiele zmiennych. Ostateczne podejście, które zostało uzyskane podczas iteracyjnego procesu poprawy skuteczności modeli wyróżnia się spośród metod dostępnych w literaturze użyciem warstw self attention oraz zastosowaniem statystyki zmian. Zaowocowało ono uzyskaniem głosującego modelu predykcyjnego, który na danych testowych uzyskał prawie dwukrotnie wyższy wynik F1 (w klasycznym sposobie oceny) wynoszący 0.2222 w porównaniu do wyniku F1 0.114 uzyskanego przez badaczy [15], którzy udostępnili ten zbiór danych. Podczas testów okazało się również, że modele typu AutoEncoder używane często w problemie wykrywania anomalii sprawują się gorzej niż modele predykcyjne.

Oba typy modeli poradziły sobie również bardzo dobrze na danych pochodzących z logserwera firmy EMCA. Modele typu AutoEncoder uzyskały precyzję powyżej 50% zarówno na podstawowym jak i rozszerzonym zbiorze, jest to precyzja wyższa niż najlepsza precyzja otrzymana na danych testowych. Model predykcyjny na danych firmy EMCA zgłosił mniej anomalii niż modele typu AutoEncoder, jednak osiągnął wyższą precyzję, która na zbiorze rozszerzonym wyniosła ponad 89%, co jest wynikiem bardzo dobrym, gdyż tylko 1 na 10 anomalii zgłaszanych przez model jest fałszywym pozytywem. Dodatkowo po przeprowadzeniu testu podstawowych modeli na zbiorze EMCA potwierdziło się przypuszczenie, że podejście rozwinięte w ramach pracy magisterskiej pozwala uzyskać lepsze rezultaty niż podejście podstawowe.

Dodatkowo w pracy rozwinięty został autorski sposób oceny modelu nazwany **Alternatywnym sposobem oceny**, który jest bardziej odpowiedni w przypadku oceny modeli wykrywających anomalie niż standardowy sposób oceny. W przeciwieństwie do tradycyjnego sposobu uwzględnia on fakt, że ze względu na zmiany zachodzące w systemie przed awarią, model może zgłosić wystąpienie anomalii na kilka chwil przed lub kilka chwil po jej rzeczywistym wystąpieniu.

5.1. Możliwości dalszego rozwoju

Jednak pomimo tego, że badania przeprowadzone w ramach pracy magisterskiej można uznać za sukces, to są aspekty, które można byłoby poprawić. Modele słabo sobie radzą z wykrywaniem anomalii w przypadku, gdy podział serii czasowej na zbiór treningowy oraz testowy wykonany został w sposób chronologiczny. Pomimo zastosowania metody Online learningu, skuteczność modeli w takiej konfiguracji była niska i zdecydowanie jest to kierunek na którym warto się skupić podczas rozwoju prac nad wykrywaniem anomalii. Kolejnym aspektem, który można byłoby rozwinąć jest zaproponowany w pracy alternatywny sposób oceny modeli używanych w zadaniu znajdywania anomalii. Ciekawym aspektem byłoby wprowadzenie mechanizmu przyznawania poprawnie wykrywanym anomaliom wag w zależności od tego, na ile minut przed krytycznym błędem została ona zgłoszona. Ewentualnie model mógłby być nagradzany jedynie za zgłaszanie anomalii przed ich wystąpieniem, natomiast zgłoszenie anomalii kilka minut po jej wystąpieniu nie powinno zwiększać liczby fałszywych pozytywów, ale nie powinno również liczyć się jako poprawna klasyfikacja anomalii. Jest to na pewno przedmiot do dalszych rozważań, na podstawie których można byłoby zaproponować ustandaryzowany sposób oceny modeli wykrywających anomalie.

Bibliografia

[1] Beggel L, Kausler BX, Schiegg M, Pfeiffer M, Bischl B (2019) Time series anomaly detection based on shapelet learning. Computational Statistics 34(3):945–976, DOI 10.1007/s00180-018-0824-9, URL https://ideas.repec.org/a/spr/compst/v34y2019i3d10.1007_s00180-018-0824-9.html

[2] Callegari C, Gazzarrini L, Giordano S, Pagano M, Pepe T (2011) A novel pca-based network anomaly detection. pp 1 – 5, DOI 10.1109/icc.2011.5962595

[3] CyberZHG (2018) Keras self-attention. https://github.com/CyberZHG, gitHub repository

[4] Davis N, Raina G, Jagannathan KP (2019) Lstm-based anomaly detection: Detection rules from extreme value theory. CoRR abs/1909.06041, URL http://arxiv.org/abs/1909.06041, 1909.06041

[5] Dubrovin MG, Gluhih IN, Karyakin IY (2020) Forecasting the server status using the triple exponential smoothing model. Journal of Physics: Conference Series 1661:012031, DOI 10.1088/1742-6596/1661/1/012031, URL https://doi.org/10.1088%2F1742-6596%2F1661%2F1%2F012031

[6] Eiteneuer B, Niggemann O (2020) Lstm for model-based anomaly detection in cyber-physical systems. 2010.15680

[7] Faouzi J, Janati H (2020) pyts: A python package for time series classification. Journal of Machine Learning Research 21(46):1–6, URL http://jmlr.org/papers/v21/19-763.html

[8] Harrou F, Kadri F, Chaabane S, Tahon C, Sun Y (2015) Improved principal component analysis for anomaly detection: Application to an emergency department. Computers & Industrial Engineering 88:63 – 77, DOI https://doi.org/10.1016/j.cie.2015.06.020, URL http://www.sciencedirect.com/science/article/pii/S036083521500279X

[9] Janus P, Ganzha M, Bicki A, Paprzycki M (2021) Applying machine learning to study infrastructure anomalies in a mid-size data center – preliminary considerations. DOI 10.24251/HICSS.2021.025

[10] Kudo T, Morita T, Matsuda T, Takine T (2013) Pca-based robust anomaly detection using periodic traffic behavior. In: 2013 IEEE International Conference on Communications Workshops (ICC), pp 1330–1334, DOI 10.1109/ICCW.2013.6649443

[11] Miz V, Ricaud B, Benzi K, Vandergheynst P (2019) Anomaly detection in the dynamics of web and social networks. CoRR abs/1901.09688, URL http://arxiv.org/abs/1901.09688, 1901.09688

[12] Moura A, Lucena S (2011) Anomaly detection using holt-winters forecast model

[13] Nguyen QP, Lim KW, Divakaran DM, Low KH, Chan MC (2019) Gee: A gradient-based explainable variational autoencoder for network anomaly detection. 1903.06661

[14] Paffenroth RC, Kay K, Servi L (2018) Robust PCA for anomaly detection in cyber networks. CoRR abs/1801.01571, URL http://arxiv.org/abs/1801.01571, 1801.01571

[15] Ranjan C, Reddy M, Mustonen M, Paynabar K, Pourak K (2019) Dataset: Rare event classification in multivariate time series. 1809.10717

[16] Ringberg H, Soule A, Rexford J, Diot C (2007) Sensitivity of pca for traffic anomaly detection. vol 35, pp 109–120, DOI 10.1145/1254882.1254895

[17] Russo S, Disch A, Blumensaat F, Villez K (2020) Anomaly detection using deep autoencoders for in-situ wastewater systems monitoring data. 2002.03843

[18] Schmidt F, Suri-Payer F, Gulenko A, Wallschläger M, Acker A, Kao O (2018) Unsupervised anomaly event detection for cloud monitoring using online arima. In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), pp 71–76, DOI 10.1109/UCC-Companion.2018.00037

[19] Tang Z, Chen Z, Bao Y, Li H (2018) Convolutional neural network-based data anomaly detection method using multiple information for structural health monitoring. https://doi.org/10.1002/stc.2296, online; dostęp 29 listopada 2020

[20] Thill M, Konen W, Bäck T (2017) Online anomaly detection on the webscope s5 dataset: A comparative study. In: 2017 Evolving and Adaptive Intelligent Systems (EAIS), pp 1–8, DOI 10.1109/EAIS.2017.7954844

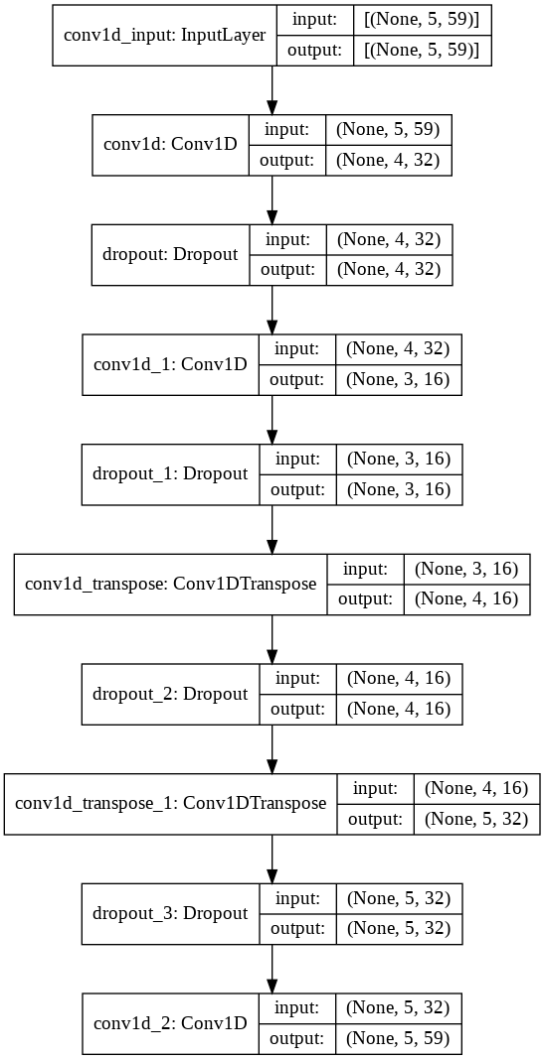
[21] Twitter (2015) Twitter advec. https://github.com/twitter/AnomalyDetection, online; dostęp 29 listopada 2020

Spis rysunków

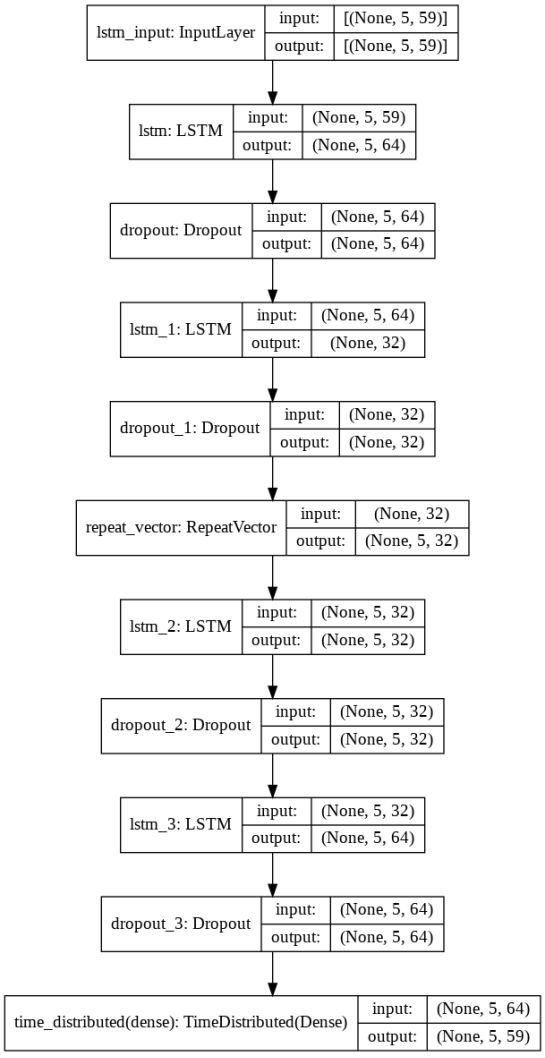
Spis tablic

6. Dodatek

6.1. Podstawowe architektury sieci

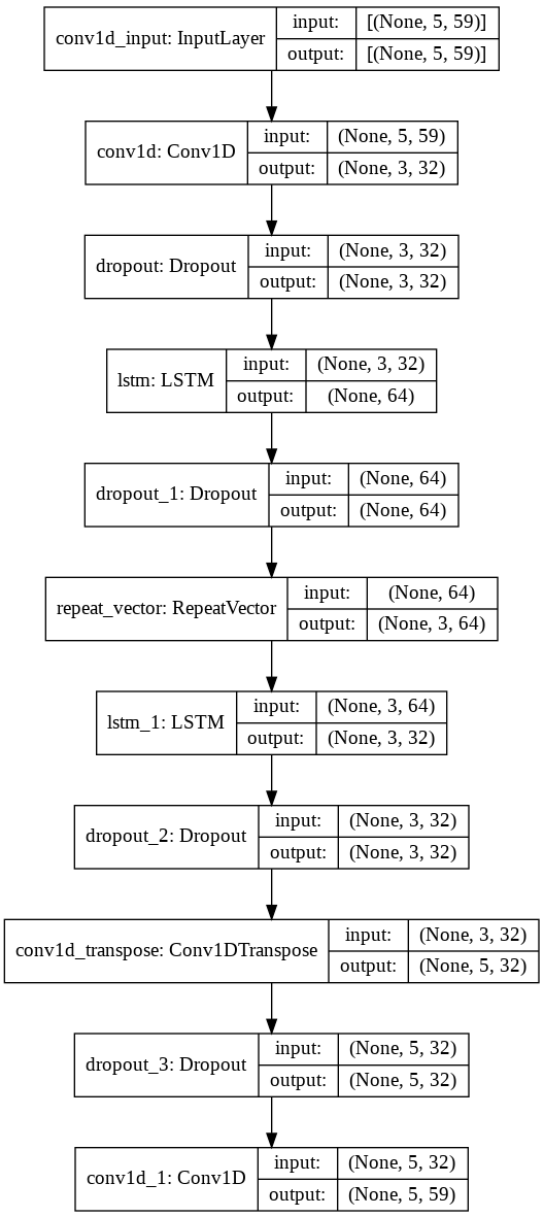


(a) Architektura sieci AutoEncoder CNN. (Opracowanie własne)

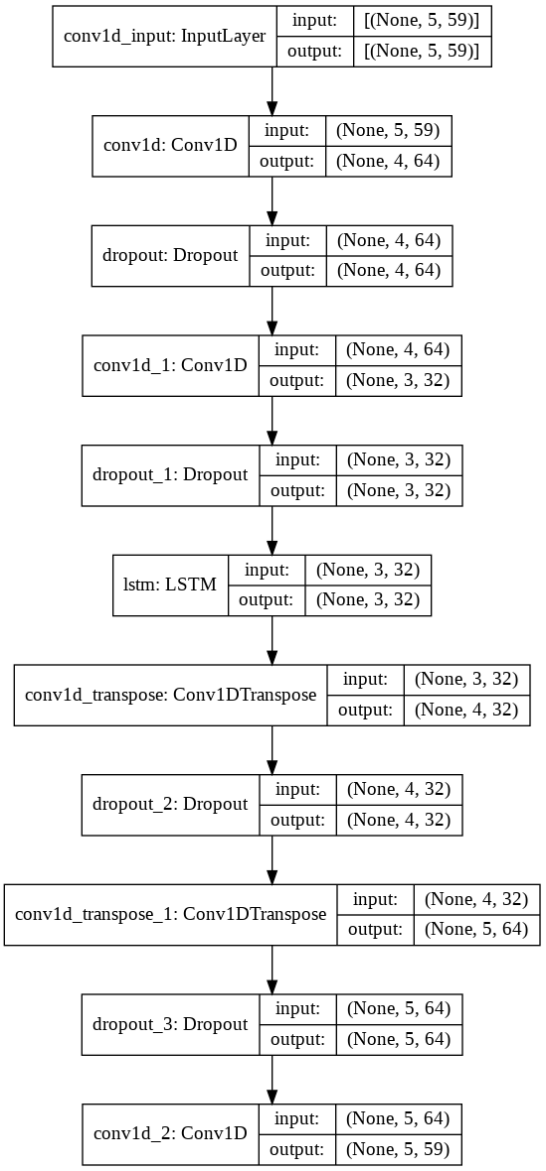


(b) Architektura sieci AutoEncoder LSTM. (Opracowanie własne)

Rysunek 6.1: Architektury sieci typu AutoEncoder

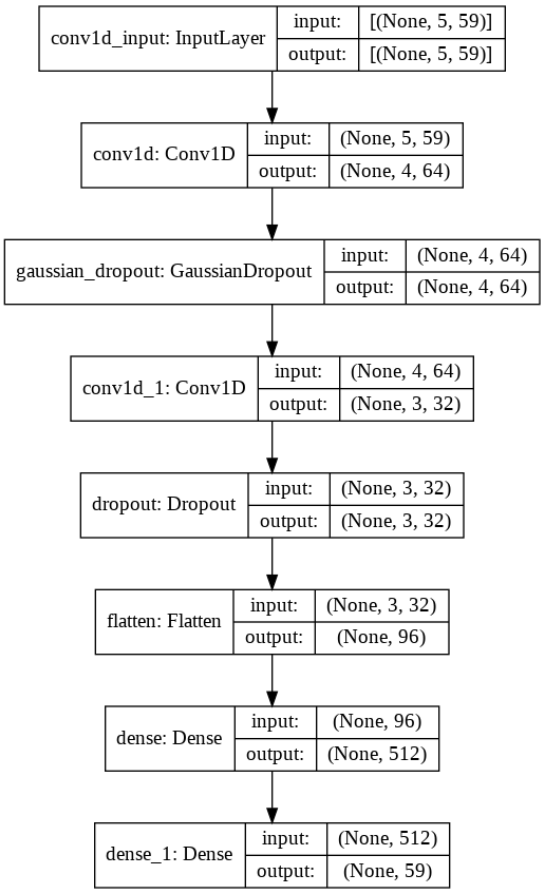


(a) Architektura sieci AutoEncoder CNN LSTM. (Opracowanie własne)

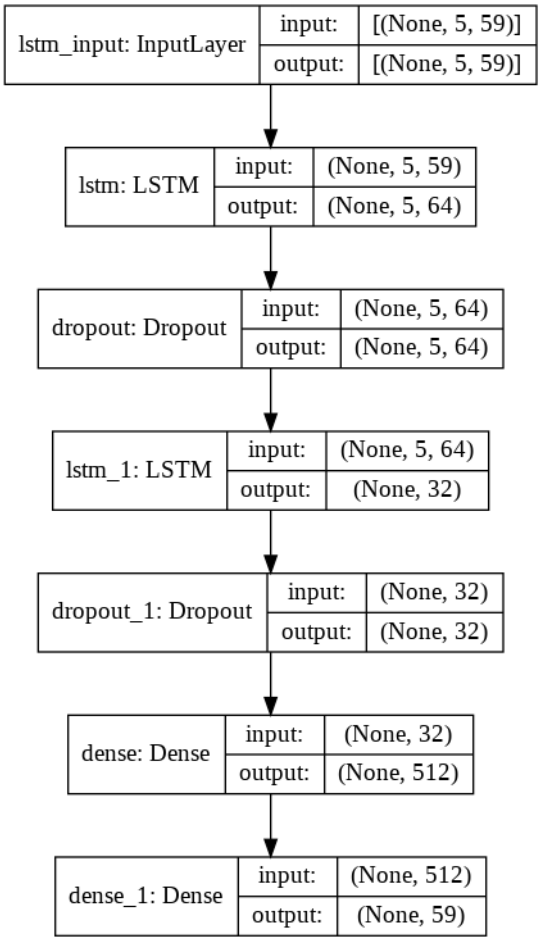


(b) Druga architektura sieci AutoEncoder CNN LSTM. (Opracowanie własne)

Rysunek 6.2: Architektury sieci typu AutoEncoder

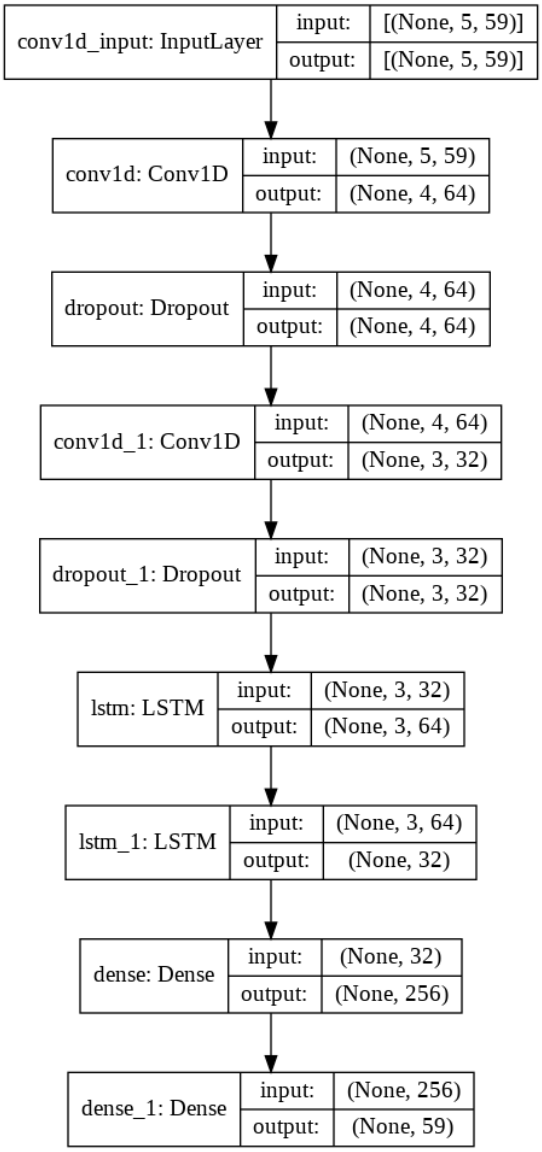


(a) Architektura sieci CNN. (Opracowanie własne)



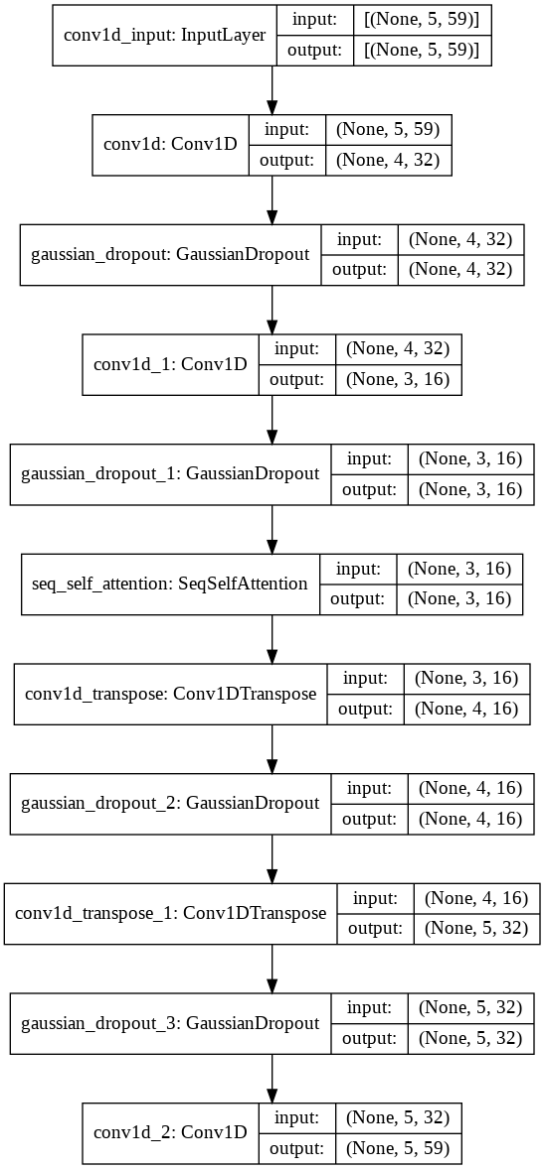
(b) Architektura sieci LSTM. (Opracowanie własne)

Rysunek 6.3: Architektury sieci przewidujących

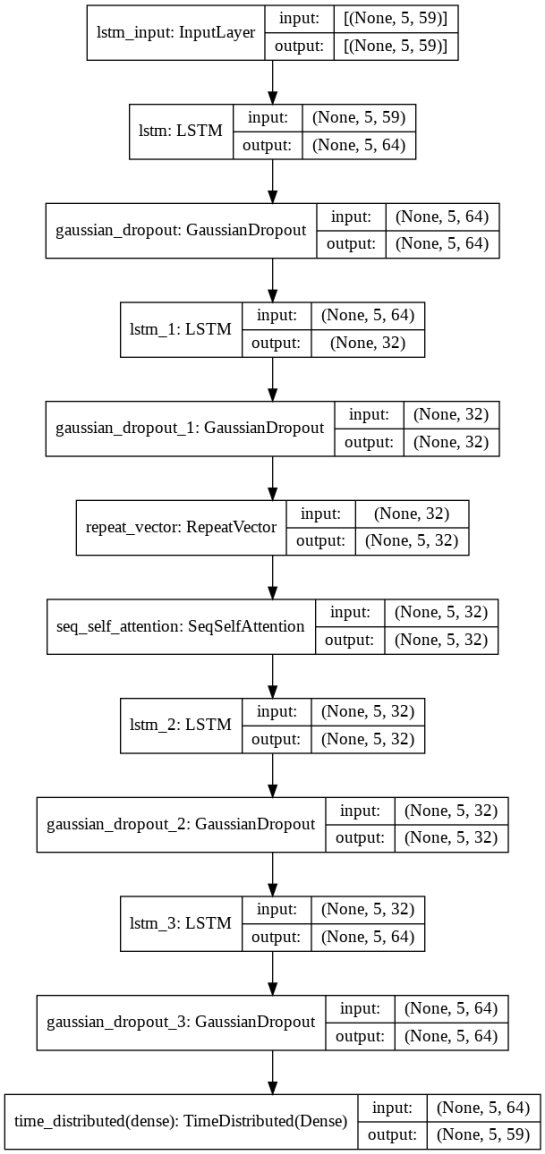


Rysunek 6.4: Architektura sieci CNN LSTM. (Opracowanie własne)

6.2. Ulepszone architektury sieci

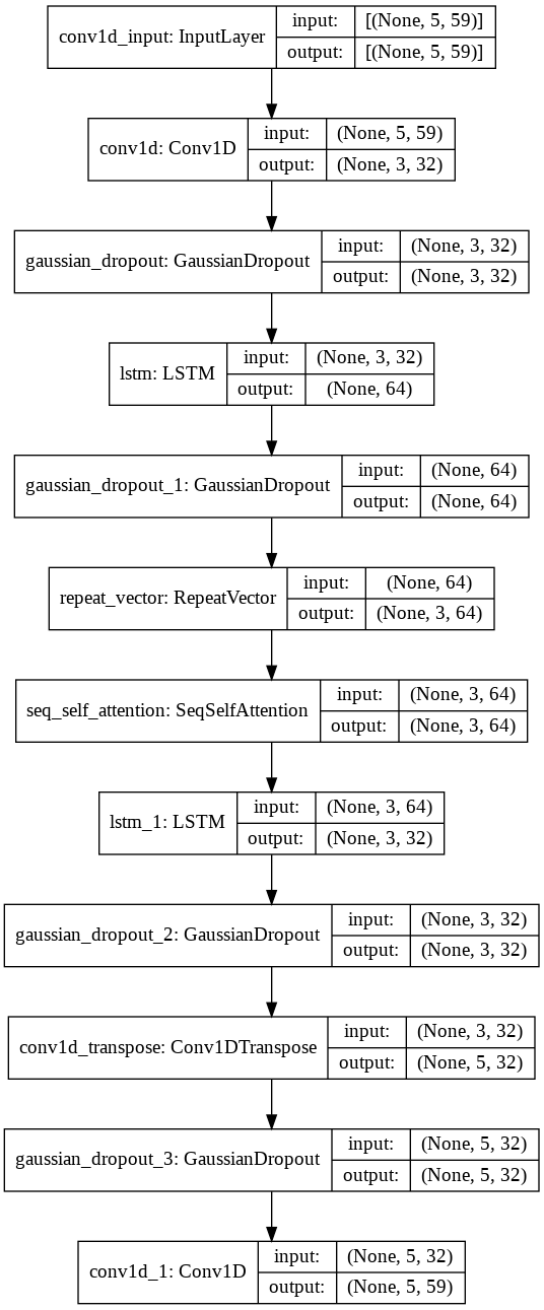


(a) Zmodyfikowana architektura sieci AutoEncoder CNN. (Opracowanie własne)

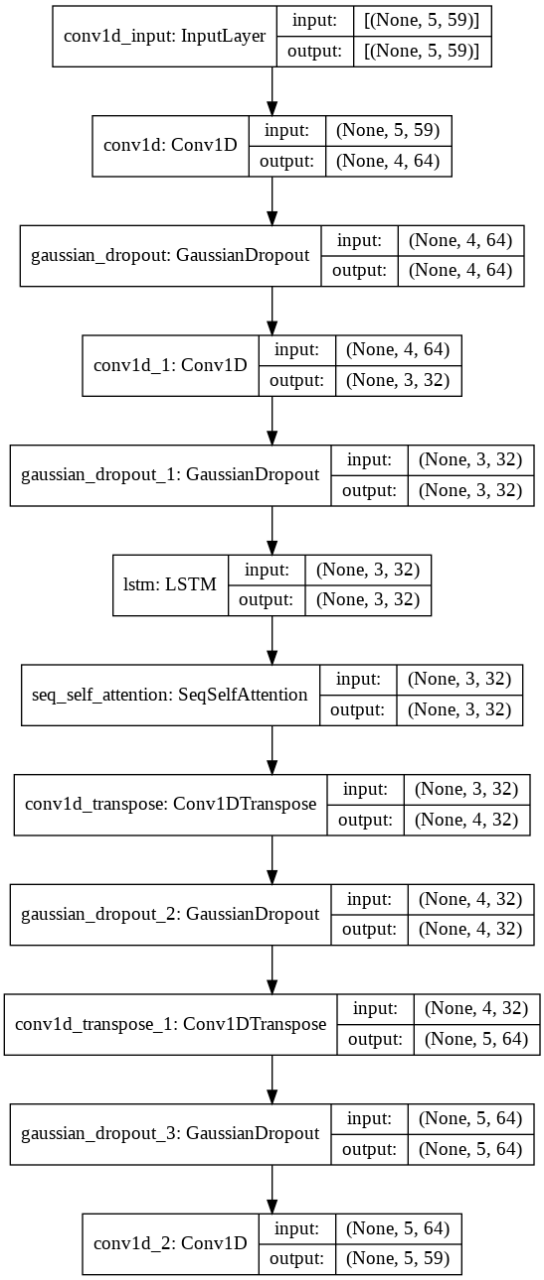


(b) Zmodyfikowana architektura sieci AutoEncoder LSTM. (Opracowanie własne)

Rysunek 6.5: Architektury sieci typu AutoEncoder

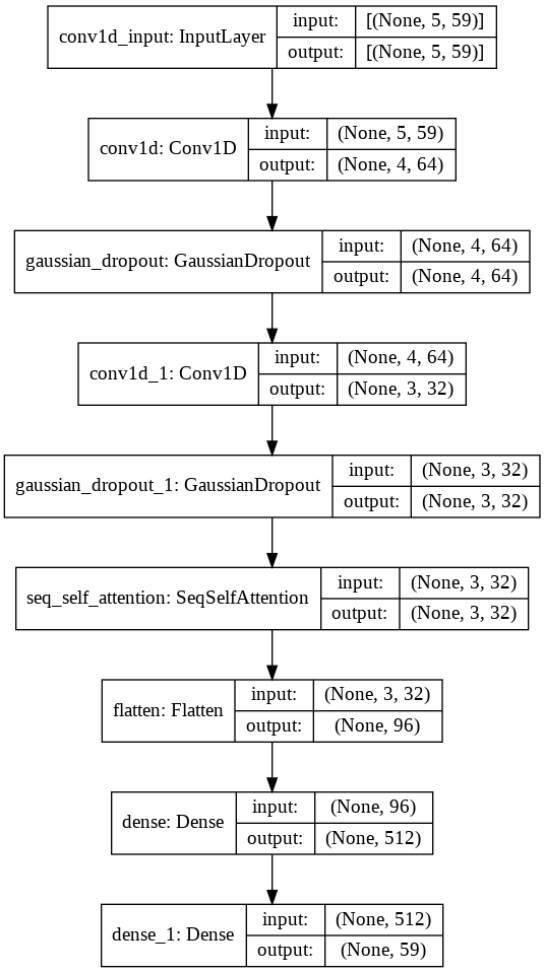


(a) Zmodyfikowana architektura sieci AutoEncoder CNN LSTM. (Opracowanie własne)

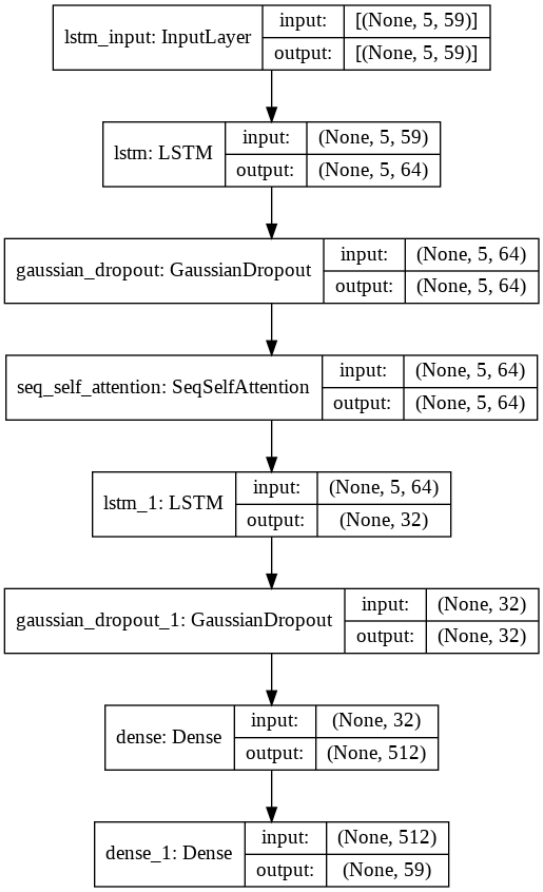


(b) Zmodyfikowana druga architektura sieci AutoEncoder CNN LSTM. (Opracowanie własne)

Rysunek 6.6: Architektury sieci typu AutoEncoder

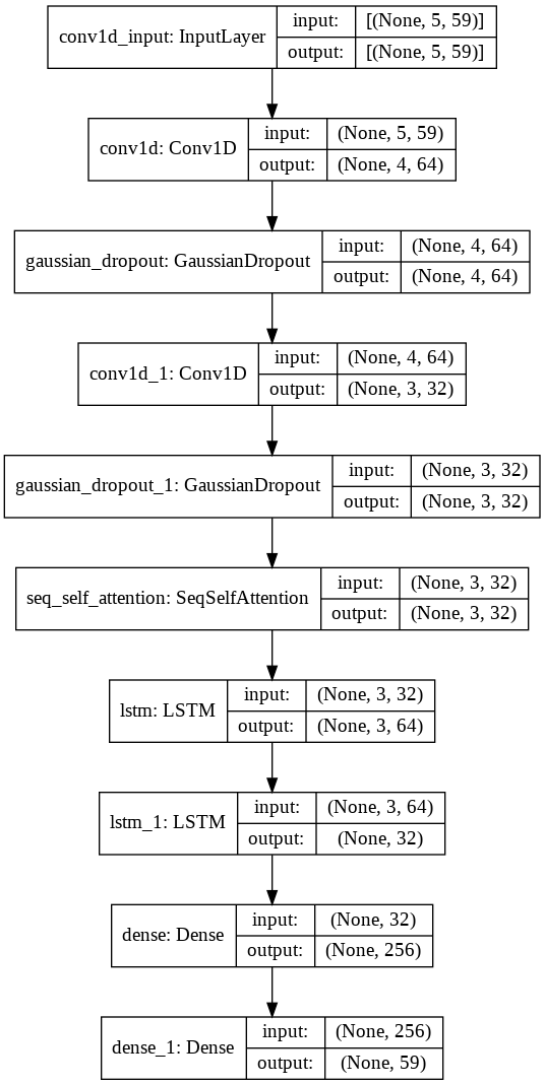


(a) Zmodyfikowana architektura sieci CNN. (Opracowanie własne)



(b) Zmodyfikowana architektura sieci LSTM. (Opracowanie własne)

Rysunek 6.7: Architektury sieci przewidujących



Rysunek 6.8: Zmodyfikowana architektura sieci CNN LSTM. (Opracowanie własne)